MANAGEMENT OF CONSTRUCTION RESOURCES

A THESIS

Presented to

The Faculty of the Division of Graduate Studies

By

Robert Lowell Preston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

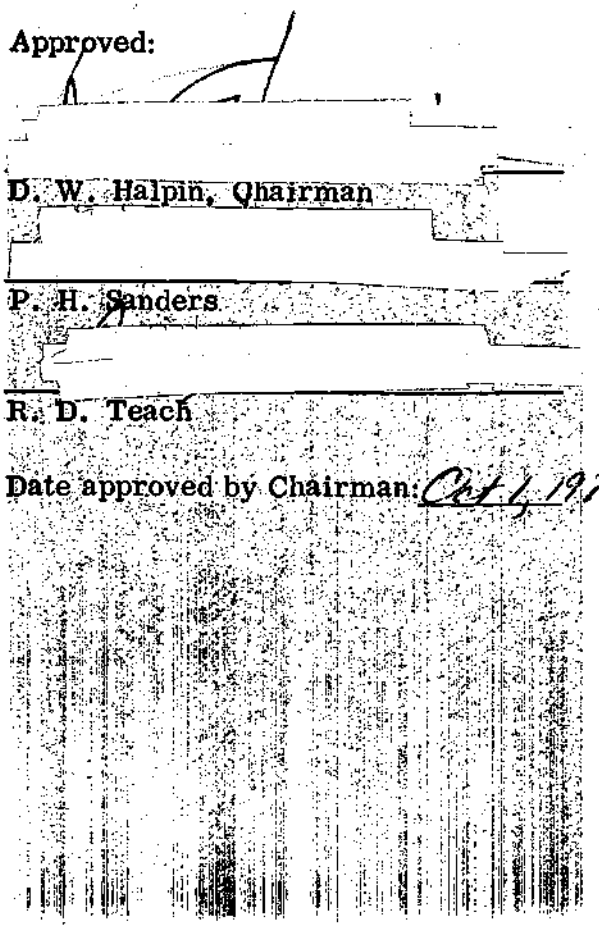
in the School of Civil Engineering

Georgia Institute of Technology

July 1976

MANAGEMENT OF CONSTRUCTION RESOURCES

Approved:



D. W. Halpin, Chairman

P. H. Sanders

R. D. Teach

Date approved by Chairman: Oct 6, 1976

ACKNOWLEDGMENTS

The generous support and guidance of my committee; Dr. P. H. Sanders, Dr. F. W. Schutz, Dr. G. N. Berlin and Dr. R. D. Teach; is greatly appreciated. I am particularly indebted to my principal advisor, Dr. D. W. Halpin who, without hesitation, gave his time, encouragement and personal knowledge on my behalf.

I am also obligated to Dr. Ross A. Gagliano for his encouragement and assistance with the more esoteric elements of machine behavior.

Finally, I am grateful to my wife Sandy and to my son Robert who have supported my efforts without question. I must also thank my daughter Teri for helping with the bibliography.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONCLUDED)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONTINUED)

LIST OF ILLUSTRATIONS (CONCLUDED)

SUMMARY

The capability of managing construction resources is expanded by adapting Discrete, Differential, Dynamic Programming to a unique and realistic formulation of the construction resource problem. The management, allocation and resource synthesis program is defined in terms of a resource requirement decision network and solved with the objective of determining the minimum cost to meet all requirements.

The need for this research is based upon a comprehensive literature survey of the current and recently current generation of resource management techniques. The survey shows that of the many techniques which have been suggested only selected heuristic procedures have gained relative acceptance by the construction industry. The lack of acceptance of rigorous, mathematical techniques can be attributed to their lack of practical value as well as limited capability in terms of assumptions which preclude practical results as well as incomplete formulations.

The management, allocation and resource synthesis program is based upon critical path logic, feasible resource quantities and resource costs which are tools already familiar to and accepted by construction schedulers. A new concept of activity float is defined as partitioned float. Partitioned float is an activity's share of total float and is valuable in that its use avoids the necessity to choose either an early start or late start schedule.

The development of relevant resource alternatives is followed by construction of the dynamic model which converts the critical path network into a serial resource requirement network and then synthesises the requirements of the different resources. The dynamic model interrelates the alternative feasible decisions on the basis of total system cost. The associated objective function using the system cost basis obviates the necessity of leveling, allocation and time-cost trade-off by effectively providing for each alternative whenever such an alternative is the most cost effective solution.

The resulting mathematical model, which requires a significant storage capability is solved by adapting proven discrete, differential, dynamic programming procedures to the synthesised resource requirement network.

CHAPTER I

INTRODUCTION

The construction industry exists in an environment that is both economically and technologically dynamic. Firms may find themselves financially unable to take advantage of cost saving developments in one time period and suffer subsequently by being unable to compete with newly developed technological innovations. In addition, the industry depends directly upon the requirements of a changing society and new technological demands for its own viability. Organizations existing with such dynamic conditions must maintain maximum control of endogenous processes to survive. While new management processes may provide the ability to control some variables, the pervasive effect of many efforts to extend construction management technology has the tendency to reduce the trust that potential users of management science have in new technology. In many cases, the science has been oversold, in others the users have been inundated with techniques that are not understood, there is no apparent need of, and some that are of no practical value.

The characteristics of the need for internal control have been and shall continue to be increasingly complex in nature. In this respect, the progressive nature of a dynamic technology continuously increases the burden of maintaining internal control in an industry which is only partially committed to the utilization of scientific decision processes. The advent of the currently widespread

recognition of a universal depletion of resources will also continue to affect the problem of internal control in a definitive manner. Not only must the construction industry contend with the increasing complexity resulting from technology, but must comprehend the necessity of meeting the requirements of new technology with a greater limitation of resource availability and a greater demand to use resources economically.

The primary endogenous variables available to the industry are its resources, men, money, material, equipment, time and space. Adaptive control of these variables will allow the opportunity to meet the requirements of the dynamic environment. Many procedures have been devised to meet these requirements. The critical path method is one of these procedures which was developed by Kelley [64] to meet the increasing need for resource control within a dynamic construction process. Kelley and Walker [64] have reviewed the development, testing and implementation of the critical path method and descriptions of the mechanics of critical path techniques have been prepared by Fondahl [44], Kelley [65], Moder and Phillips [80], O'Brien [82], Shaffer, Ritter and Meyer [100].

While Kelley originally intended CPM to be more inclusive of resource considerations, early applications of critical path techniques for the most part do not treat the application of manpower and other resources required to perform the different work elements of a construction project. Sufficient resources were either assumed available to meet the schedule or the resource problem was ignored entirely. Consequently, while both planning and scheduling are

basic to internal control the failure to incorporate adequate consideration of the various resources required by a project nas limited the effectiveness of critical path techniques for planning, but has more seriously left the technique without a vehicle by which it can be mapped to a schedule.

Since the basis of this study is the alternative employment of resources required by a project's activities and their consequent impact on the accomplishment of a project, a discussion of the resources which concern a construction project is in order.

Resources

A construction project as an entity is an arrangement of material according to some preconceived plan. To effect this specific arrangement, additional material, manpower and equipment are generally required along with some element of time as well as an element of space. In a practical sense, each of these elements share a common resource--money. Resultingly, construction resources are of five categories with an additional basic resource which is common to each of the types as shown by Figure 1.

Each type resource can be controlled with the exception of time which can only be used [35]. While the controllability of four of the resources allows their treatment as variables, the range of control is limited by two physical aspects. First, every resource has an upper boundary which is determined by its availability to the user. In this respect, resource availability is limited by the geographic project location, the time required to obtain or create the resource, the amount of the resource or components of the resource that exist

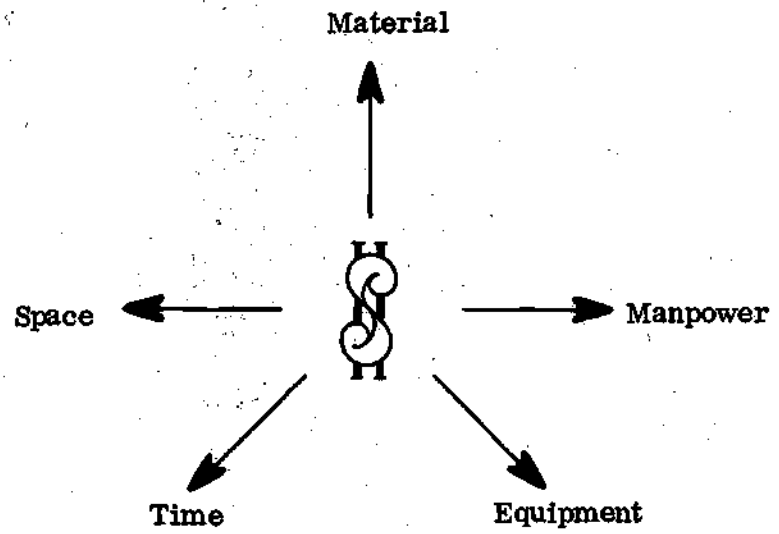


Figure 1. Categories of Resources.

and any limitation on the cost of the resource. The second aspect of resource control has both an upper and lower boundary that depend upon the environment of the project as a whole as well as the physical conditions under which the individual resource will be used. The lower boundary is frequently greater than a single resource because some resources can only be used in pairs or mixtures. For example, a concrete bucket must necessarily be accompanied by a means of lifting the bucket; a pusher tractor is often required to load one or more scrapers; a carpenter may require a laborer to help lift lightweight beams into position; a mortar carrier frequently serves more than two masons. As an upper boundary, there is a limit on the number of resources which can effectively be utilized on a particular activity or in a particular work space.

Before any variation of resources can be effected, it is necessary that the individual resource requirements be mapped to a specific time segment of a project. The critical path network readily serves such a function by providing

precedence relationships from its logic and a time frame from the early start, late start, early finish and late finish times which result from the critical path calculations. The resource profile such as that shown by Figure 2 can be prepared from critical path information. A resource profile shows the level or cumulative amount of an individual resource required within an element of time.

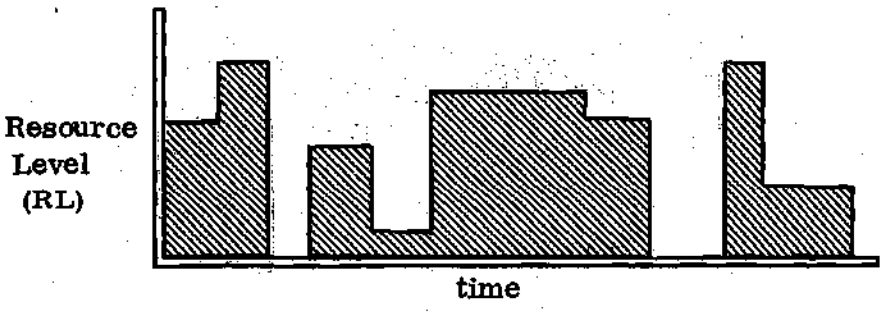


Figure 2. Resource Profile.

In addition to the resource profile, a resource annotated, early start bar chart similar to Figure 3 can be drawn to show the resource level required by an activity as well as any free float available to the activity.

In a scheduling context there are only two basic operations which may be imposed upon a resource:

a. The amount of a resource applied to a job can be changed.

b. The time of application of the resource to a job can be changed.

The subsequent effect of these two basic operations can then serve to accomplish what has been termed by Dunne [35], Margenthaler [71], and Richards [92] to be either Critical Resource Analysis (CRA) or Cost Duration Analysis (CDA).

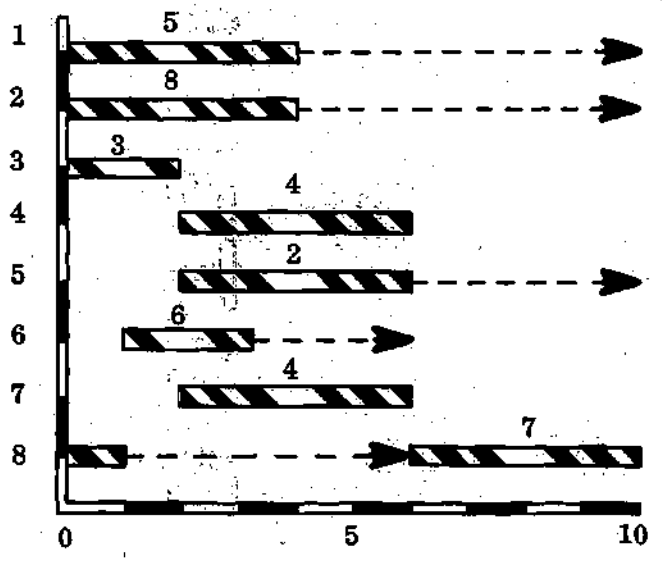


Figure 3. Resource Annotated, Early Start Bar Chart.

Critical Resource Analysis

The CRA approach attempts to find the best project schedule when resource limits have been imposed. The process is accomplished in either one or two ways: (a) by minimizing the changes in the application of individual resources over time (leveling) or (b) minimizing the maximum resource level required for the entire project duration (resource scheduling).

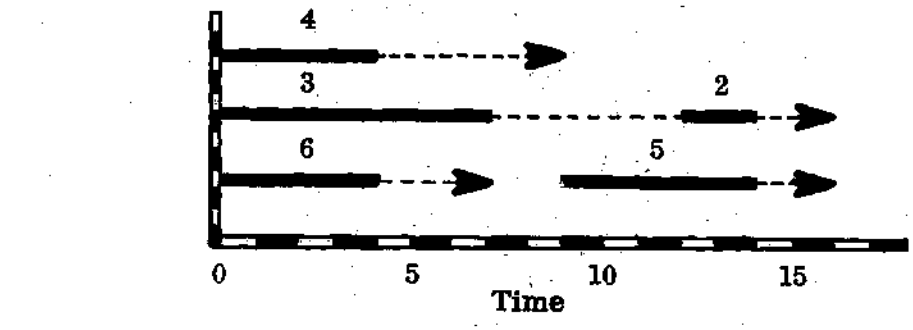
Leveling

Within respective limits, resource leveling is cost effective since the expense of moving a resource from one project to another is avoided if the resource is used continuously until its requirement has been exhausted. In some situations, consistent utilization avoids periods of non-productive time when the resource must remain at the initial project site. In addition, manpower has other cost advantages when used at a consistent level: It may be just as

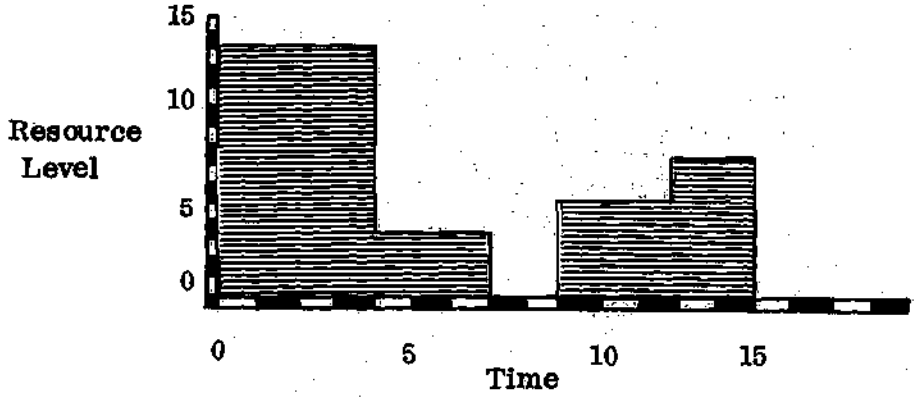
impractical to hire ten masons on Monday, let them go on Wednesday, then hire five new masons on Friday, as it is to let all ten remain on-site for two days with no work assignment. Resource leveling can be accomplished by shifting non-critical activities within any time allowance between the late finish time and the earliest, early start time of succeeding activities. This free float time (or possibly total float time) can be used to reduce the resource level by scheduling the activity in a time frame which will minimize the total resource requirement with a constant project duration. Figure 4 illustrates this shifting process within respective free float times to minimize resource levels.

Efforts to level resources or to minimize maximum resource levels may not be independent of time. The process of leveling can further be accomplished by allowing total project time to vary. In the constant time case (leveling), resources can only be shifted within existing free float or total float time with the resulting schedule yielding a minimization of the changes in resource utilization over time. Alternately, time may be allowed to vary and resources assigned to activities to conform to a maximum constraint on the resource level. Unfortunately, unless other constraints are added, the lowest resource limit always produces an optimum schedule and renders the solution functionally intractable.

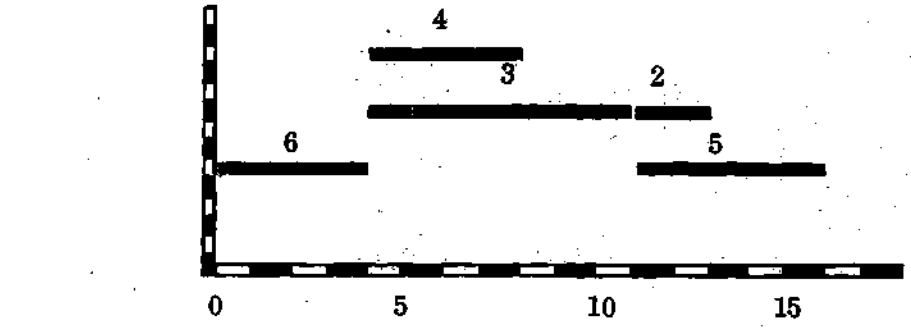
The procedure of assigning the accomplishment of an activity to a variable time frame subject to a maximum resource constraint and allowing project time to vary has also been termed "allocation" as well as constrained resource scheduling.



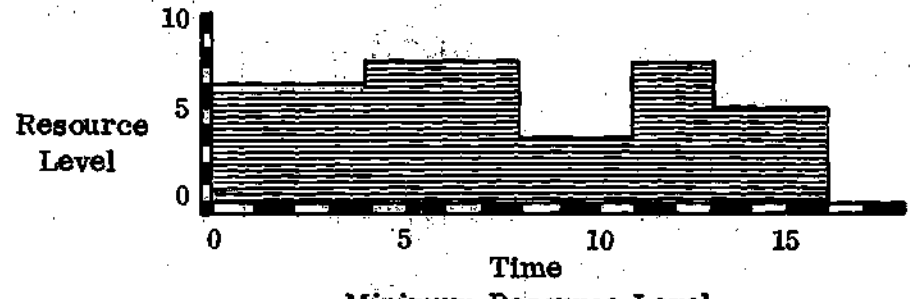
Free Float Resource Requirement



Early Start Resource Profile



Resource Requirements After Shifting



Minimum Resource Level

Figure 4. Resource Leveling within Free Float.

Cost Duration Analysis

The duration of many construction activities can be altered by applying different levels of resources. Within applicable limits, an activity requiring forty man-hours can sometimes be accomplished by one man working forty hours or forty men working one hour. The relationship between resource level (RL) and duration may not be linear and in fact may assume a variety of forms including non-linear, discrete, discontinuous, or a combination of the alternatives as shown by Figure 5.

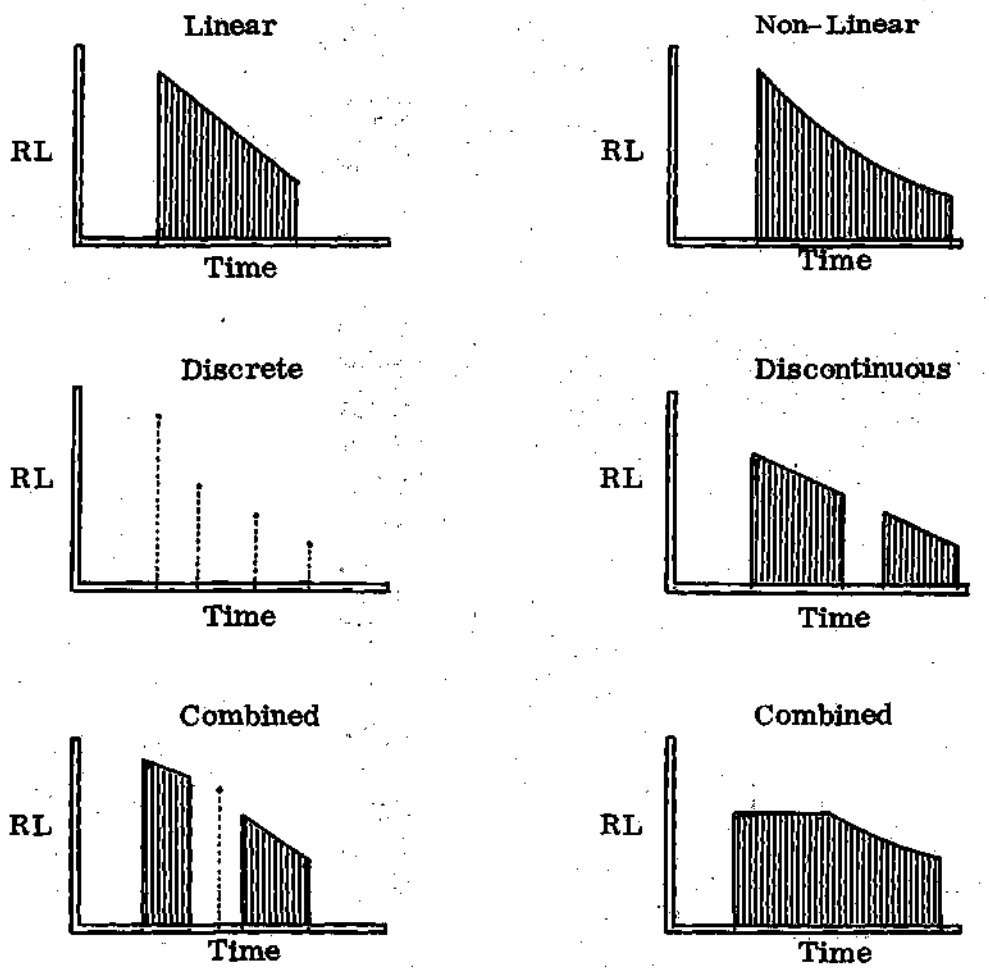


Figure 5. Resource/Time Combinations.

The time/cost trade-off situation results from the consideration of the cost of using different resource levels and the cost of the time subsequently changed by changing resource levels. It is noted that time/cost trade-off has also been considered part of resource allocation [35]. Figure 6 shows a series of activities whose durations can be varied with the subsequent reduction in project duration that can result from reducing the duration of the activities on the critical path.

Between the options of resource operations there exist some interesting relationships. First, a minimum time schedule with constrained resources is coincident to a schedule leveled to the same resource level. This concept is illustrated by Figure 7. This is true since minimum time requires maximum resource utilization. Secondly, a time/cost trade-off analysis may be of interest in either the constrained or non-constrained resource case. The return of leveling the cash resource must be balanced not only by the expense of the leveling itself, but also by the expense of earlier expenditures of funds and the cost of maximum negative cash flows.

Recognition of the limitations of the critical path techniques without consideration of resource application is widespread. To remedy this limitation, numerous techniques have been proposed and are surveyed in Chapter II. As the survey shows, the techniques are still lacking in one or more of the aspects of optimality, simplicity, construction applicability, computational feasibility, or severely limiting assumptions.

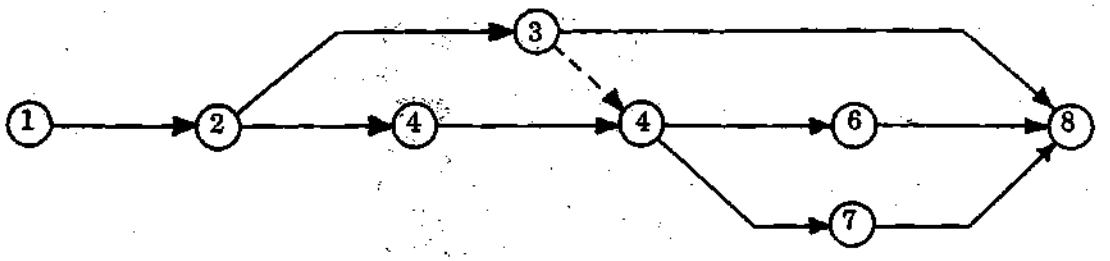


Figure 6A. Example Network.

Activity	Normal		Duration Reduction	
	Duration	Cost	Time Available	Cost
1-2	6	100	2	20
2-3	9	300	4	80
2-4	4	150	1	30
3-4	0	0	-	-
3-5	7	200	2	30
4-6	8	500	5	125
4-7	2	200	1	50
5-8	1	300	0	-
6-8	6	100	1	20
7-8	5	400	3	90

Figure 6B. Time/Cost Data.

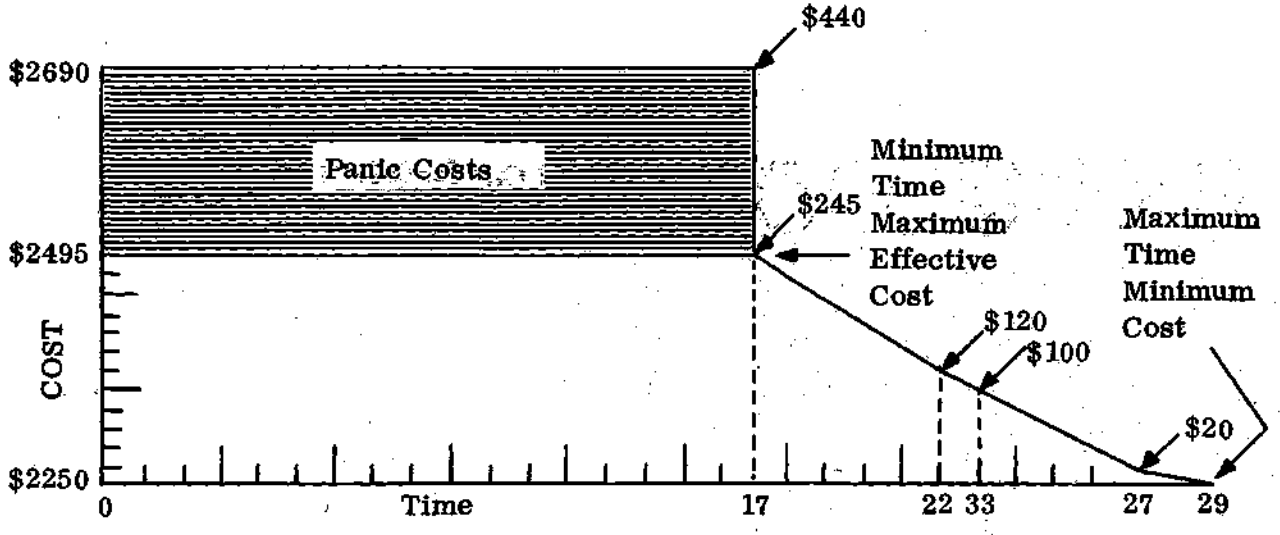


Figure 6C. Example Cost Duration Curve.

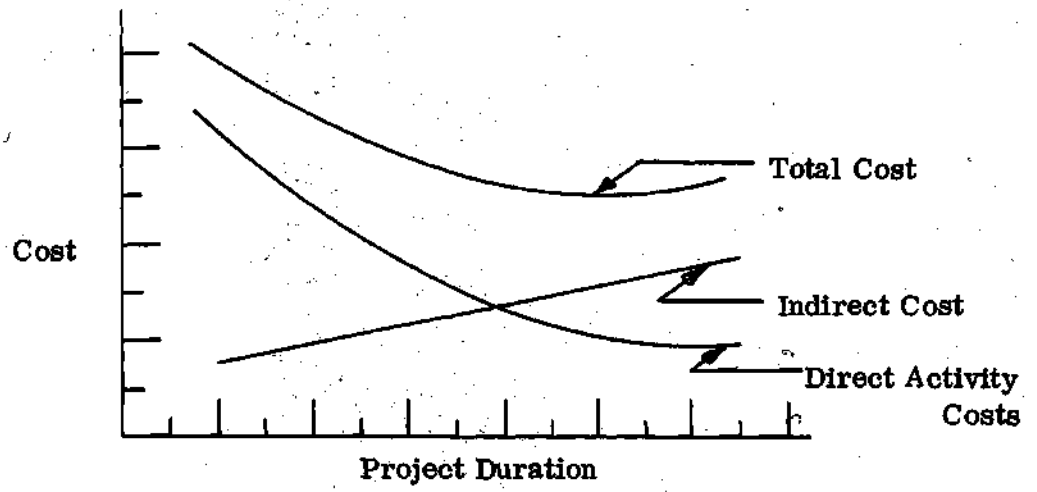


Figure 6D. Minimum Cost Duration.

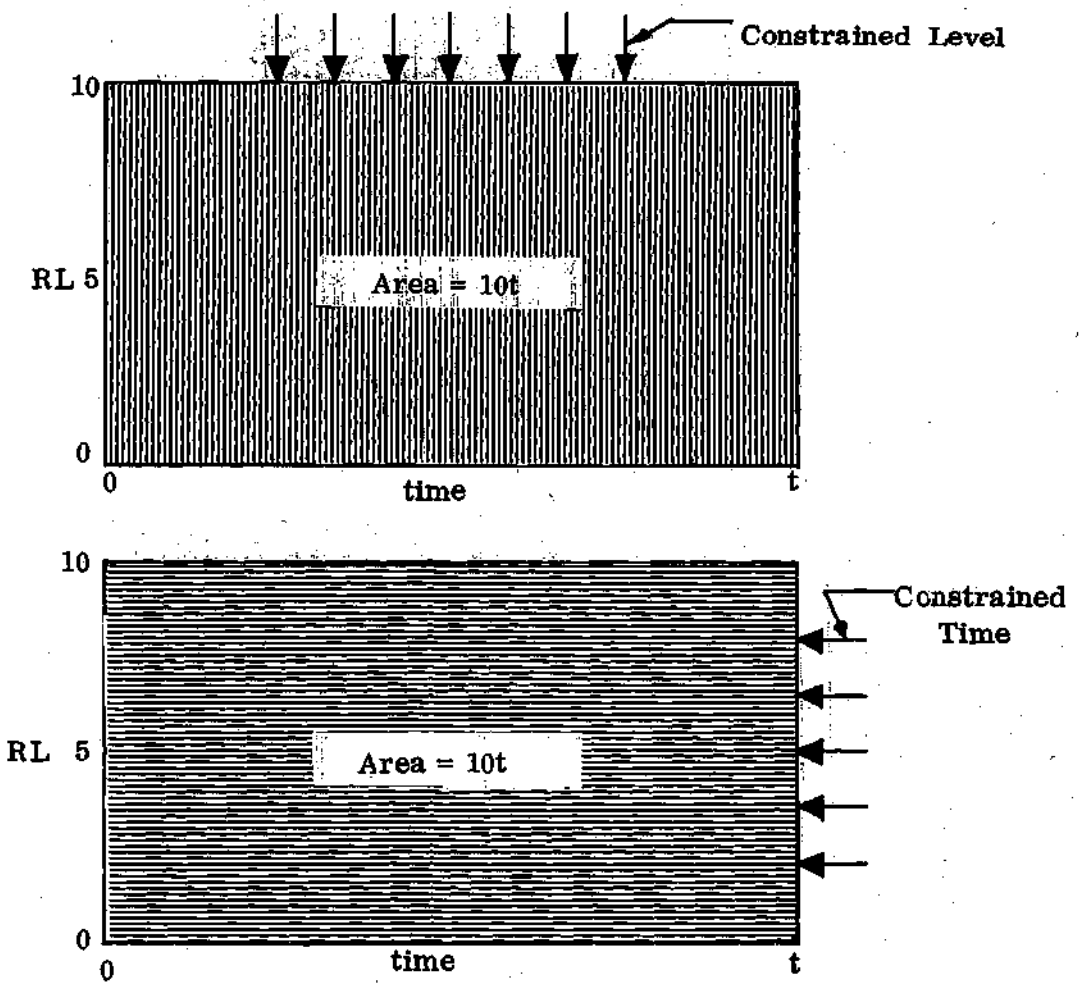


Figure 7. Constrained Level/Constrained Time Diagram.

Purpose of Research

The purpose of this research is to develop a mathematical model based on a critical path schedule which is of practical use to the construction industry for controlling the resource scheduling process in an economical manner. The practical model will incorporate variables of primary importance in the scheduling problem while variables of secondary value will be consciously traded for the measure of simplicity required to render the model useful to the construction industry. Specifically, the model will provide a feasible resource constrained and economical selection of resource level and time of resource application to result in the completion of the total project.

The development of the research will consist of a survey of existing methodologies relating to the resource management problem, Chapter II; Dynamic Model Development, Chapter III; The Proposed Method of Solution by Dynamic Programming, Chapter IV; A Demonstration of the Methodology, Chapter V; Evaluation of Results, Chapter VI; and finally the Conclusions and Recommendations for Future Work, Chapter VII.

CHAPTER II

NETWORK MANAGEMENT MODELS--STATE OF THE ART

Introduction

The solution of construction resource management problems is primarily concerned with the scheduling of resources for construction work. However, the potential of a practical solution to the resource allocation problem is not unique to the construction industry. The continually increasing demand for more economical use of scarce resources is particularly evidenced by the volume of literature addressing allocation within general industrial network applications as well as the job-machine scheduling problem. Consequently, the general solution process overlaps the industrial job sequencing or loading problem. While the industrial application is frequently much less complex, some of the techniques employed are not only similar, but also imply the possibility of unusual formulations of the construction management problem. Due to this overlap, appropriate shop scheduling models as well as scheduling models designed primarily for other than construction use will be reviewed along with the construction resource management models.

The solution of construction and related resource management problems has been approached in a variety of ways. Many procedures are based on selected heuristic rules, while others utilize linear, integer, zero-one and dynamic programming, as well as selective adaptations of solution methods

for other problems.

Heuristic Methods

Procedures based on heuristic rules are widely used primarily because they offer a means of obtaining feasible resource schedules for large, complex projects. Heuristic procedures are also popular for planning schedules because the user can rationalize low expectations for an optimal schedule by considering the uncertainties already known to be associated with the activity durations of a planning schedule. In addition, the speed of computation of simple heuristic algorithms by computer often makes possible an effective simulation process by trying several different algorithms and selecting the best schedule.

Because heuristic procedures are so firmly established, it is useful to review the general way by which a heuristic schedule is generated. Most procedures based on heuristics begin with a CPM early-start schedule. The CPM is then divided into time periods and each time period is examined to see if the resource availability has been exceeded. If a resource limit has been exceeded, each activity within the time period is compared to a heuristic such as "minimum float first" to determine which job should be scheduled first without exceeding the resource limit and subsequently which job should be delayed. Methods proposed by Spivak [102], Moder and Phillips [80], Dike [33], and Brand, Meyer and Shaffer [15] are of this type. Some heuristic rules in frequent use include:

(a) Early start and late finish first

(b) Minimum early start first

(c) Minimum late finish first

(d) Shortest duration first

(e) Minimum total float first

(f) Maximum duration first

As an example of a heuristic schedule, Figure 8 displays a bar chart derived from a CPM early start schedule. Figure 9 is an early start resource profile derived from the bar chart, Figure 8. Should the resource level of Figure 9 be limited to ten and a "shortest job first" rule be used, the 13 day schedule of Figure 10 would result.

Alternately, Figure 11 illustrates that a better schedule can be derived by inspection [26].

Because the rules generally in use do not lend themselves to a description of a unique activity, there are frequent cases where the criteria for scheduling are met concurrently by more than one activity. Consequently, heuristic rules may require two or more additional heuristics when the first produces no difference between activities.

Heuristic techniques existed as early as 1956 when Galbreath [47] proposed a heuristic for leveling resources within their free floats. Even though this was an early effort, Galbreath recognized the adverse costs associated with changes in resource levels such as hiring and firing costs, cost of moving equipment on and off site, lack of efficiency of new crews and the cost advantage associated with the more competent tradesmen who gravitate towards longer periods of sustained employment.

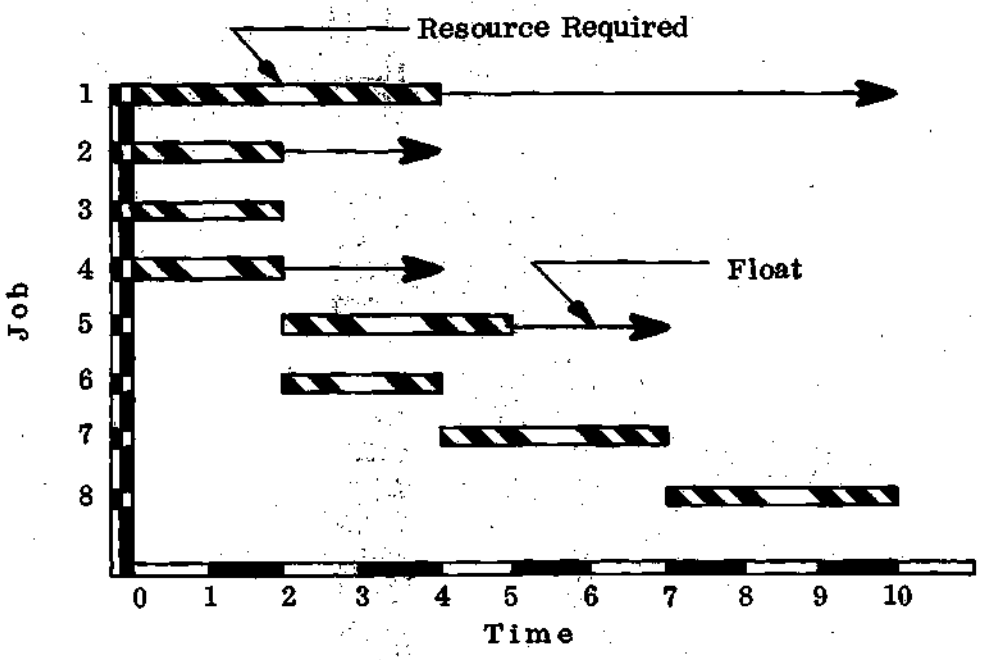


Figure 8. Resource Bar Chart.

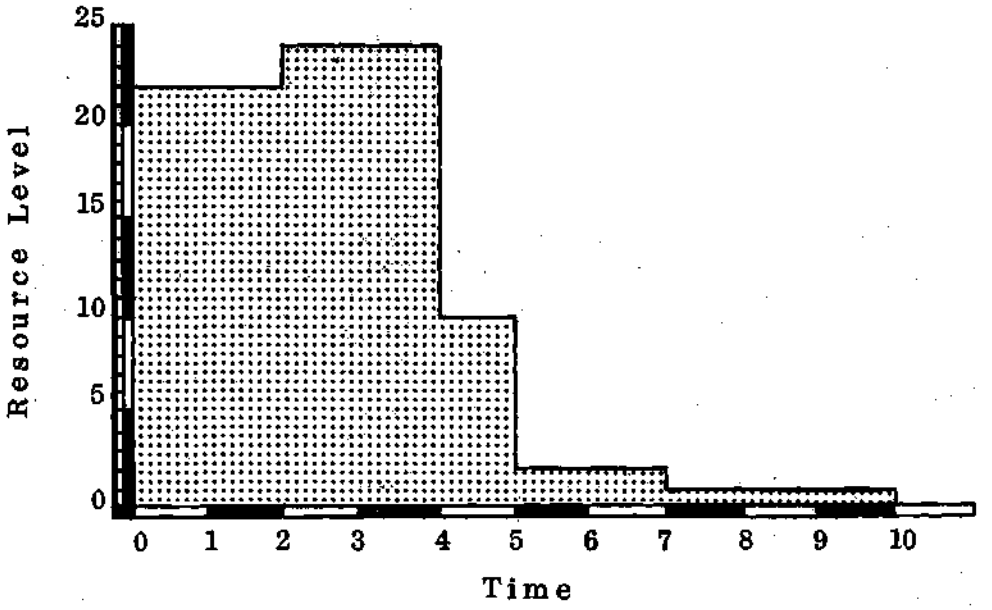
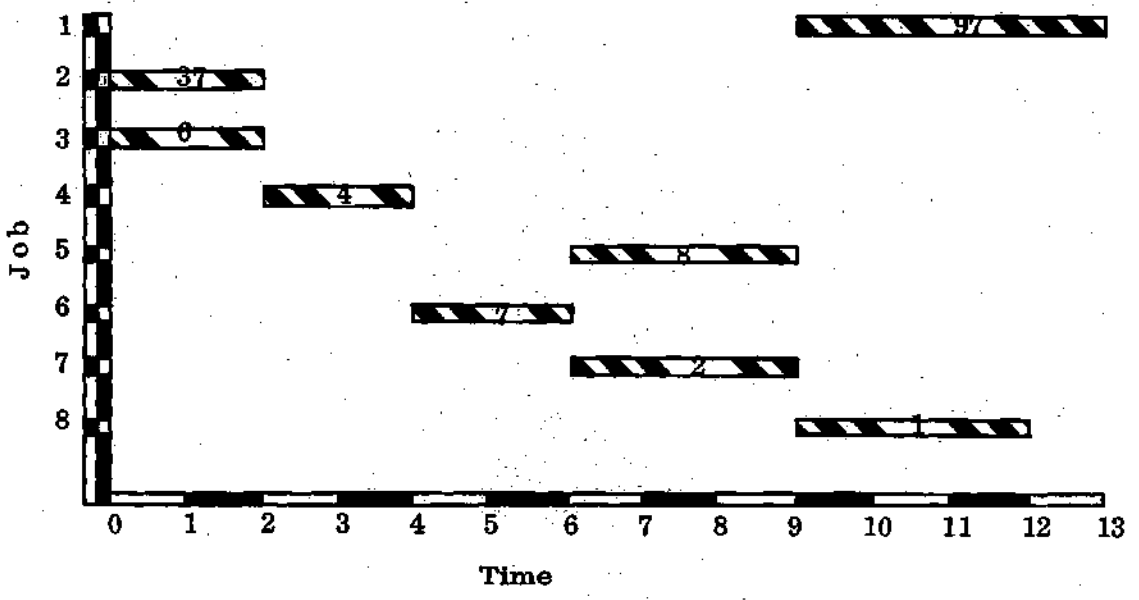


Figure 9. Resource Profile Diagram.



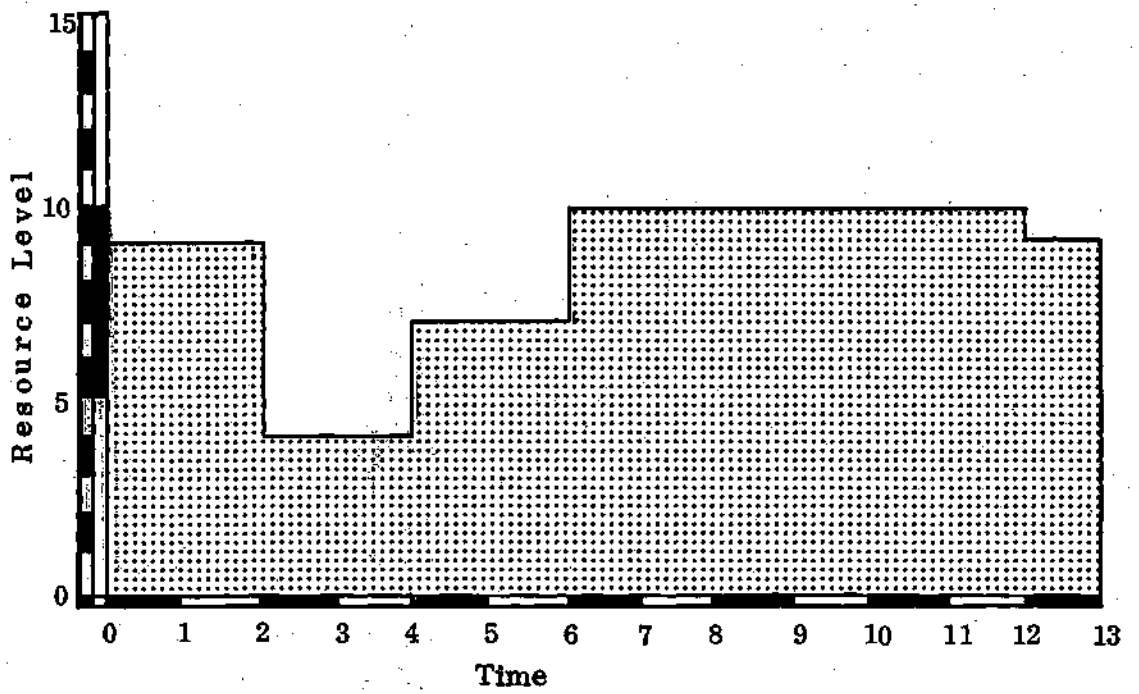


Figure 10. Shortest Duration First Bar Chart and Resource Profile.

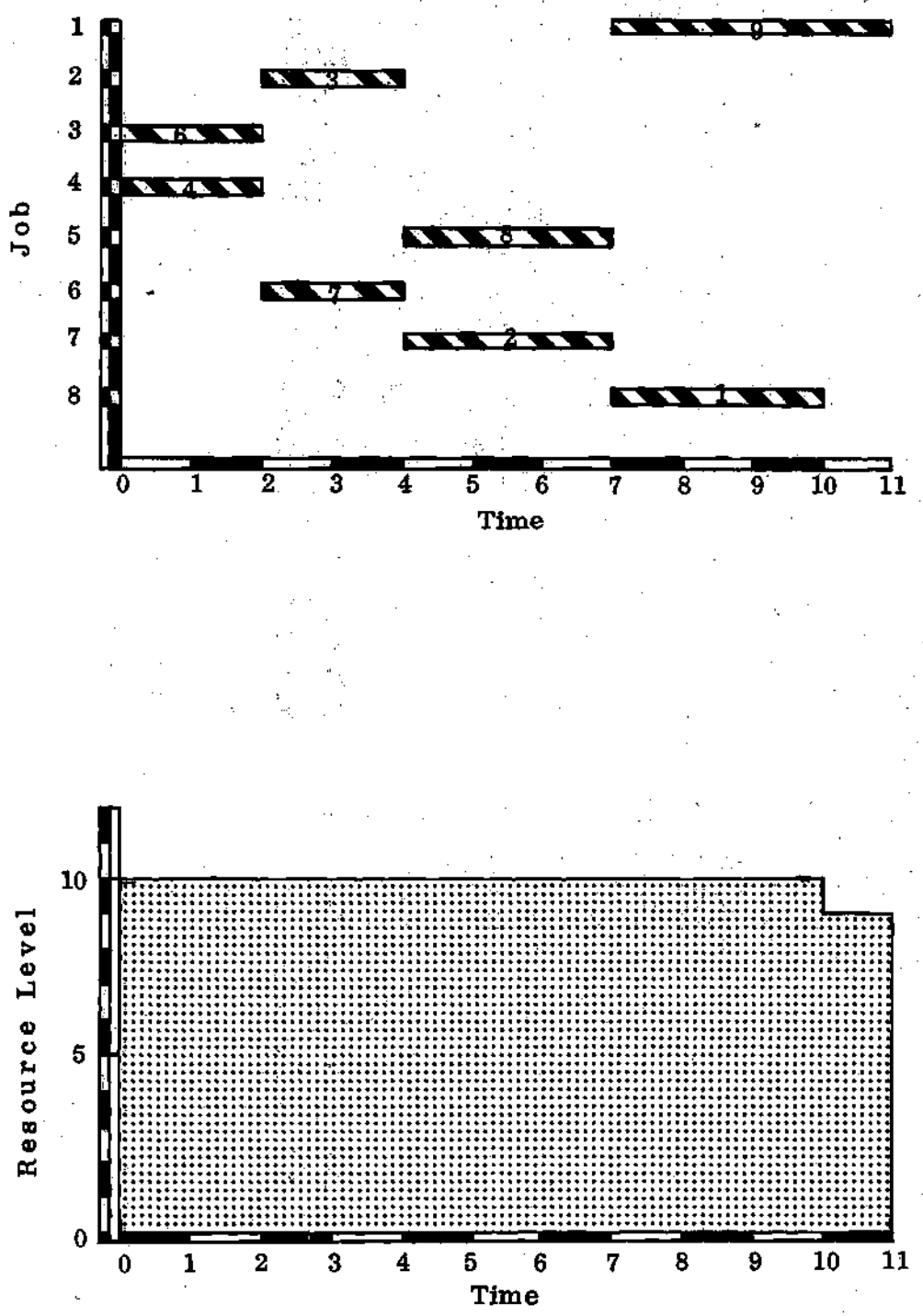


Figure 11. Bar Chart and Resource Profile Derived from Observation.

Kelley [64] proposed two algorithms involving serial and parallel methods of determining schedules having minimal lengths within resource limitations. The algorithms allowed work on an activity at intermittent times and provided for varying duration of an activity by increasing or decreasing the amount of resource applied. The serial algorithm scheduled an activity on its early start time provided sufficient resources were available. When resources were not available the activity was delayed for consideration in a later time period. As opposed to the serial method, the parallel method scheduled several activities at one time. Activities considered for scheduling were those whose predecessors had been completed and those for which adequate resources were available. The success of both Kelley's algorithm and the criterion of minimizing the sum of squares of all resource levels suggested by Burgess [16] in 1962 are primarily dependent on the ordering of activities prior to scheduling. DeWitte's [32] proposal for resource leveling was based on the absolute value of deviation from an average resource demand over the project duration.

In 1964 Brand, Meyer and Shaffer [15] evaluated six heuristics:

(a) The manual method

(b) Sequencing by float

(c) Dee's Manpower Optimization

(d) Shirley's Critical Path

(e) Man-scheduling

(f) Resource Scheduling Method

Brand et al. [15] concluded that heuristic techniques offer the most reasonable

methods for solution of the constrained resource problem and appear to favor the Resource Scheduling Method as the most satisfactory of the heuristics.

Baker and Shaffer [3] applied staged decision theory with total float heuristics to the network problem to solve the critical resource scheduling problem. The method was restricted to one operation per activity and did not produce a conclusive result.

One of the first multi-project heuristic models was developed by Mize [79] in 1964. Working from a CPM network, a procedure to minimize total project slippages was formulated. As a simplification, Mize assumed all activities were independent and limited application to one resource. The resulting heuristic scheduling rules concluded that parallel sequencing procedures were better than the serial procedure. Although the procedure was lacking in mathematical rigor, the impact of mobilization costs, demobilization costs, resource movement costs and a time associated with each cost element was considered in conjunction with each activity. In another multi-project work, Levy, Thompson and Weist [68] give a multi-project procedure that was designed to reduce the most recent peak resource usage.

Weist [111] recognized that CPM and PERT schedules assume unlimited resources and in fact when resources are limited the time schedule produced by either method is likely to be considerably different from the schedule which is actually possible when the effect of the lack of resources has impacted on the productivity of different phases of the project. To counteract this problem, Weist introduces the concept of critical sequence and discusses its implications for

project scheduling. After developing a technological ordering of jobs, Weist discusses a set of heuristic rules which shift jobs locally in descending order of their early finish times, late start times and size of crew to create the critical sequence. The schedule can then be shortened by applying greater resources to the critical sequence with a subsequent attempt to optimize a total cost function.

In a later work, Weist [112] developed the SPAR programs as management tools for decision making. The SPAR system uses a two-phase heuristic interactive, search procedure. First, the minimum resource level is established and then increased. Second, a resource level for each activity that causes all activities to start at their early start time is determined. Then these levels are decreased. Weist's return function considers overhead costs, lengths of schedules and average resource costs. Sunaga [103] used a somewhat similar heuristic procedure of shifting activities within feasible ranges to minimize project duration, but did not demonstrate optimal solutions.

Fendley [43] takes the position that analytic approaches are unsuccessful in solving scheduling problems of practical size and states that little has been achieved by interactive methods. Based on this position Fendley tests seven alternate heuristic decision rules and determines that the Minimum Slack First rule approaches optimality, although no rule proved unanimously best. The research was conducted on eight mock projects having up to 20 activities. The beta distribution was assumed to hold for all activities and random activity completion times were generated by Kahn's Rejection Method [60]. To aid in calculation, a PERT completion time was substituted for due dates and the heuristics

judged on the basis of their resultant slippage past the PERT completion time.

Paulson [85] proposed a heuristic which enables resource allocation, leveling, time-cost trade-offs and cash flow analysis to be considered as interdependent parts of the comprehensive problem. While the methodology is heuristic, the understanding of the practical aspects of the construction resource problem is especially notable. In particular, the recognition that float and critical path concepts do not retain their conventional meaning is pointed out by Paulson. Additionally, Paulson implies a transition from an activity logic network to a resource oriented network by applying the critical sequence concept.

Bennett [10] approached the scheduling problem from a resource constrained viewpoint and attempted to schedule a project in the shortest possible time. The heuristic rule proposed, considered an activity's variable time profile, variable resource levels within an activity's duration and the allowability of a resource mixture. In addition, the capability of fixing an activity's duration was included within the algorithm. As a supplementary factor, the order of assigning available resources was simulated through variable listings, none of which includes any priority assignment of activities such as free float.

Zaloom [113] also recognized that the minimum time required to process a project is dependent more on resources available than the minimum network time computed. The solution method is a summation of resources required in each time period and a subsequent comparison with the maximum resources available to establish a lower bound on minimum project duration. The solution is produced for both early and late start scheduling and attempts to reschedule

activities where the resource limit has been exceeded even though the late start schedule will also be exceeded. Zaloom proposed four lemmas which seek minimum project duration under various resource loading constraints. Zaloom did not consider optimum resource levels, optimum time nor adaptability to multi-project requirements.

Margenthaler [71], in one of the more recent efforts, has developed the heuristic of early start and late finish (ES+LF). Working from a CPM model Margenthaler used an objective function of minimizing project duration for resource constrained projects. The work compares the ES+LF rule to several other methods currently in use and concludes that the ES+LF rule is at least as good as competing rules. The utility of the procedure is restricted by assuming a fixed resource level, fixed activity durations and the lack of consideration of indirect costs.

Outstanding among the varied aspects of heuristic developments are the elaborate, computer routines which have been produced by organizations for both in-house and proprietary use. Martino [72, 75] has developed two programs for Control Data Corporation. These programs are called MAP (Multiple-Resource Allocation Procedure) and use a method termed "fixed leveling." The basis of the procedure is to reduce or eliminate all stand-by and overtime variances from the maximum resource limit. Other programs such as RAMPS (Resource Allocation and Multi-Project Scheduling) [21, 22, 66, 81] are marketed by Control Data Corporation and probably use a similar routine. Each method schedules constrained resources using its own heuristic criteria. Benson and Sewell [11] recently reported a commercially available CDA program capable of

handling 40,000 activities and Hollander [54] described another program capable of using a variety of assumptions relative to activity cost functions.

The details of most of the proprietary routines are unknown due to the lack of publication. However, the capabilities of some of the more important programs have been summarized by Davis [26].

(a) CPM-RPSM ("Resource Planning and Scheduling Method") CEIR, Inc. The heuristic is capable of handling a maximum of 8000 activities per project, 4 resource types per project, 26 resource variables and constraints, allows job splitting and activity start-finish constraints.

(b) MSCS ("Management Scheduling and Control System") McDonnell Automation Co. Multiproject capability for 26 projects, 18,000 activities, 12 resource types per activity and includes project costing reports.

(c) PMS/360 ("Project Management System") IBM Corp. Capable of handling 225 projects, 32,000 activities, and 250 resource types. Includes auxiliary costing and updating features.

(d) PPS IV ("Project Planning System") Control Data Corporation Multi-project capability with a maximum of 2000 activities per project, 20 resource types per project. Allows costing, overlapping jobs, progress reporting and resource leveling with a fixed duration.

(e) Project/2, Project Software, Inc. Handles 50 networks, 32,000 activities and several hundred resource types. Includes cost analysis and network generation features.

While in practice heuristic procedures have enjoyed maximal success in comparison with other methods, even sophisticated scheduling techniques can not guarantee an optimal or near-optimal schedule. In fact, it is impossible to predict how near a heuristic schedule is to the optimal as well as which heuristic or combination of heuristics will produce the best results for a given problem. For this reason alone, the heuristic procedures are considered intractable even though some heuristics will produce a more realistic schedule than would otherwise be available.

Programming Models

Linear Programming

Kelley [62, 63, 64, 65] and Fulkerson [46] developed similar linear programming applications for the cost-duration problem of a project which is considered to be made up of an ordered series of separate jobs. Using these assumptions (a) that each activity has an independent function, (b) that each job has a known normal and crash completion time, and (c) that the interval between crash time and normal time is a linear, continuous, convex and non-increasing function of time, the problem was solved for all feasible time intervals by formulating the dual of the primal linear programming problem and using the Ford-Fulkerson [45] network-flow algorithm to create the project cost curve which is piecewise linear and convex. The objective function of this parametric linear programming formulation is the minimization of the sum of all activity costs. The constraints are the precedence relationships between activities and

the limits on activity durations resulting from the application of variable resources.

The assumption that the cost of each activity can be approximated accurately by a continuous convex cost function as shown by Figure 12 is not universally applicable to construction problems. The continuous function implies that a discrete time exists for an activity's duration at any point on the curve. In the construction problem, the function is more likely discontinuous, having intervals where no feasible completion time exists.

Meyer and Shaffer [77, 78] have also presented extensions to the CPM to determine the minimum project cost for each feasible project duration. Formulations are developed to describe operation cost-duration relationships that are non-increasing and bounded, piecewise linear, and continuous but nonconvex; bounded but defined at only discrete points; and bounded but discontinuous. A zero-one variable is used and the formulation will solve a minimum cost project duration problem. The general CPM model was formulated as follows: Project P composed of arrows (operations) (i,j) where $i < j$. y_{ij} represents a bounded variable of duration i,j, and c_{ij} is the cost of y_{ij}. c_{ij} is expressed as a function of y_{ij}. The following states these properties:

$$d_{ij} \leq y_{ij} \leq D_{ij} \tag{1}$$

$$c_{ij} = f_{ij}(y_{ij}) \tag{2}$$

$$y_{ij} \in P, \text{ for all } (i,j) \tag{3}$$

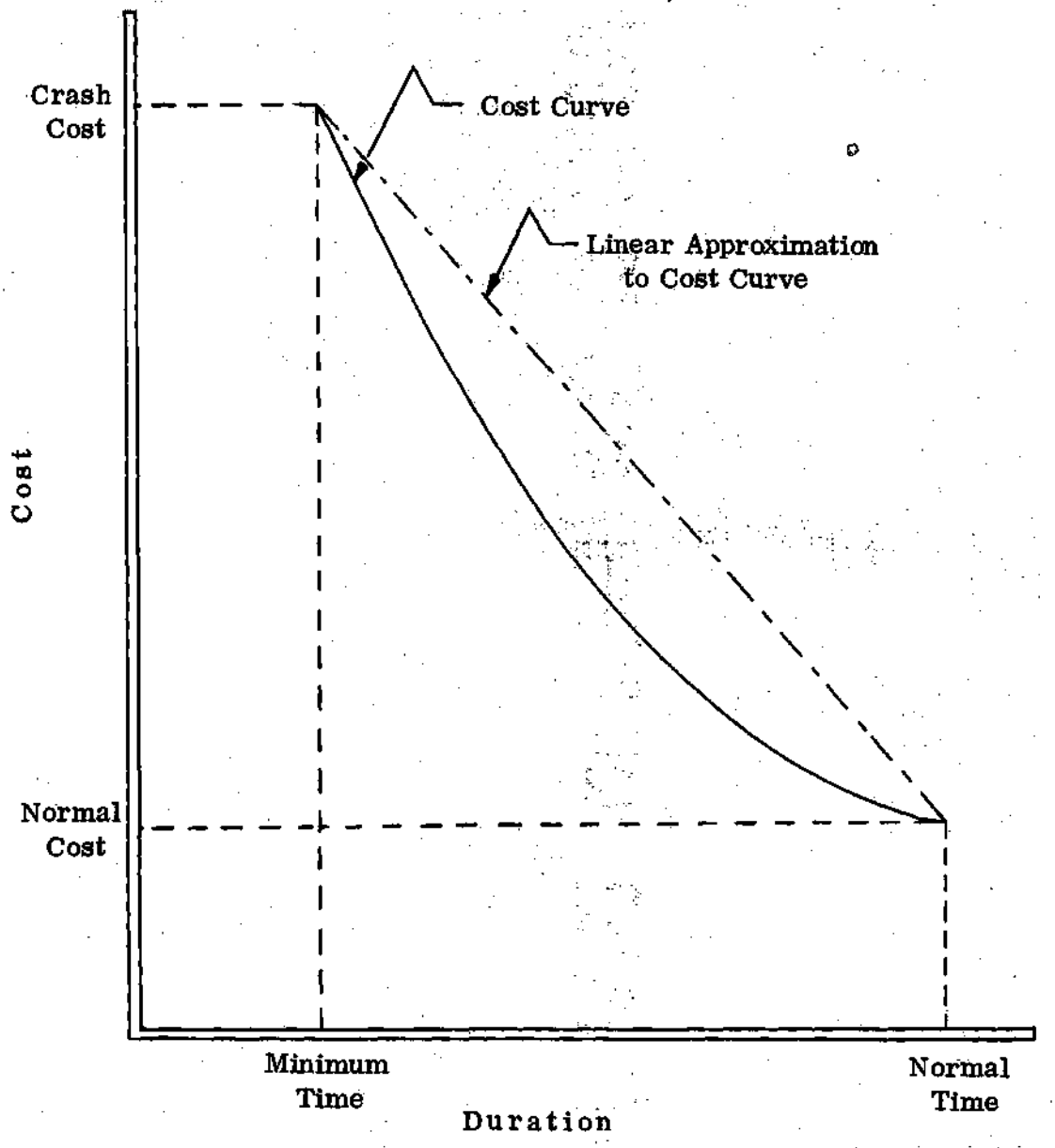


Figure 12. Linear Approximation to the Project Cost Curve.

d_{ij} is the shortest possible duration and D_{ij} is the longest possible duration. The cost of Project P is the sum of the costs of all operations. λ is the sum of the operations which are on the critical path.

A mathematical program is then established to minimize the project cost for each feasible project duration (λ).

The value λ that is chosen will determine the value of the project cost function.

Formulations were then developed to improve upon both Kelley and Fulkerson's requirement that each operation duration cost curve be bounded, nonincreasing, continuous, piece-wise linear, and convex. The formulation allows one point operation duration cost curves as a special case. While the approach is more realistic than the network flow approach, the number of variables and constraint equations increase rapidly with network size and a project of approximately fifty activities is the maximum which can be solved by this procedure [77].

Brand, Meyer and Shaffer [15], and Meyer and Shaffer [77] also proposed an improved integer linear programming technique to solve the constrained resource problem. The technique established precedence relationships between activities requiring the same resource and applied 0/1 integer variables to establish minimum project duration. While the procedure is exact, it is also limited by computational capacity.

Balas [4, 5] approaches the optimal resource scheduling problem using an additive algorithm dealing only with variables restricted to 0, 1 values. For

each problem that is created by a branch, the variables are partitioned into three classes: (1) those set equal to 0, (2) those set equal to 1, and (3) those which may be either 0 or 1. If the cost coefficients are positive, the solution obtained by setting all variables in class (3) to 0 is feasible and optimal. If that solution is non-feasible the sum of the cost coefficient of class (2) variables plus the smallest cost coefficient of the class (3) variables describes a lower bound for the optimal solution. A specific class (3) variable is transferred to class (1) and the same variable is transferred to class (2) to effect branching.

In 1967, Burton [17] used the basic CPM formulation of a linear programming problem to develop a resource scheduling model with the objective of minimizing costs. The major and defeating assumption is that the resources assigned over time form a linear, concave function.

Elmaghraby [39] constructed the optimal activity duration as a linear programming problem with linear constraints using the Kuhn-Tucker theorem to identify the global minimum. The solution depends on zero float in all activities for optimality and is consequently unrealistic since activities with lower bounds on their resource levels frequently cannot be extended to zero float.

In 1969 Pritsker, Watters and Wolfe [91] proposed the solution of the limited resources multiproject scheduling problem with a 0/1 linear programming approach. The formulation was again limited by the efficiency of the 0/1 computer codes available at that time.

Thangauelu and Shetty [107] presented an improved version of the Bowman-White zero-one integer programming formulation of an assembly line balancing

problem. The formulation was based on Geoffrion's [48] algorithm which was shortened and modified to give a more computationally efficient solution.

Jewell [58, 59] investigated a scheduling problem to optimize total time by dividing a single critical path activity into segments with separate completion times. An algorithm for determining the allocation that minimizes total project duration was proposed. The cost of interrupting the activity was neglected and a linear programming application used to obtain an initial optimal solution. Then, minimum flow techniques were applied to find the dual minimum flow solution and a new optimal solution. Jewell's approach is limited to continuous, convex activity functions and involves the use of quadratic programming.

Richards [92] utilized a more efficient 0/1 programming algorithm [56, 70] to solve an integrated CRA/CDA problem for a short range planning horizon. Consideration was given to resource mixes, movement of resources between jobs, resource mobilization costs and project completion costs. The objective function expressed the total system cost over a planning horizon. While the system appears to function adequately, the complexity of problem formulation is great and its use is limited to a planning horizon which may not include the complete construction project.

Glover and Klingman [49] have reported the development of a primal code for the solution of network and network related problems which is 100 times faster than state-of-the-art LP codes. While no details of the specific capabilities of the code are available, it is reported that the incorporation of an integer variable capability for the transportation problem is under development.

Should practical application of this code prove efficient, new possibilities for resource scheduling will be available through linear programming.

Dynamic Programming

Butcher [19] describes a limited application of dynamic programming to a series of activities and special cases of dual activities to obtain an optimum time-cost completion strategy. Allocation of the basic resource, money, is made serially to produce a minimum cost time. While it is noted that the paper was intended to show no more than a general application, the deficiencies in applicability to the general resource allocation problem are extensive, especially in the area of the inability of the method to accommodate networks that cannot be added either serially or in parallel such as Figure 13.

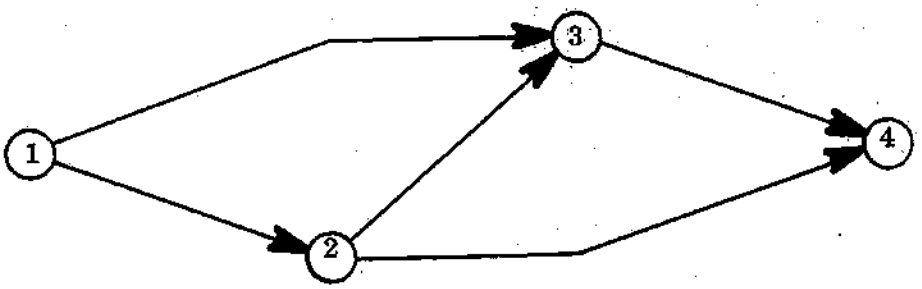


Figure 13. A Network That Cannot be Added Serially or in Parallel.

Petrovic's [87] dynamic programming formulation of the resource allocation problem maps the predecessor-successor relations of activities into a set of transformations where the state variables are the amounts of work necessary to complete the activities. The resource levels assigned to each activity

constitute the decision variables. The problem is then considered as a discrete multistage decision process and solved as a resource leveling problem.

Due to the high dimensionality of the formulation Petrovic suggested decomposing the project in time to produce subprojects which can be optimized individually and will produce total concordance when a price correcting algorithm is applied.

As a second means of handling high dimensional problems, Petrovic proposed an iterative approximate approach to improve successive feasible policies until the value of the criterion decrement becomes less than a specified tolerance.

Elmaghraby [38, 41, 42] also approached the problem of sequencing different jobs, each having a unique penalty cost, from a network model which is equivalent to a dynamic programming approach. The network is established by denoting the set of all jobs numbered in ascending magnitude of due dates.

Some job must be performed last at a given penalty cost. Subsequently a new job is selected to be performed last. Forming of subsets is continued until the empty set is reached. Then the nodes of the network correspond to the sets of the dynamic programming formulation and optimization is equivalent to the determination of the shortest path in a directed acyclic network. The approach assumes strictly convex and linear cost functions.

Rubinovitch [97] suggests a dynamic formulation of an inventory problem correlating excess inventory with a time factor which is needed to absorb random delays to activities. The provision of more inventory is translated to the cost

of adding additional resources to reduce the time required to complete a series of activities. The distribution of activity probability functions is assumed known and parametric procedures are used to construct the optimum starting times for linear, continuous, convex problems of limited scope. The application of inventory techniques is unique and provides a reasonable fit to the resource allocation problem even though the size of problem that can be handled is too limited to be of practical value.

The optimal sequence of two problems is investigated by Litsios [69] with respect to allocating one reusable resource to n tasks and allocating a non-reusable resource to n tasks. The solution technique utilized is a combination of dynamic programming and combinational analysis techniques. The two problems are shown to be respective duals.

Combinatorial Methods

Schrage [98] considers the constrained resource case where an activity may be preempted. The resources are limited to one per type and an algorithm is presented to minimize the finish time by implicit enumeration. The dominant activity attribute reduces the branches explored. Optima for up to 35 activities is found with reasonable ease.

Ellwein [37] outlined an implicit-enumeration scheme that, while maintaining a low storage requirement, allows the same flexibility in backward branching as is commonly experienced in forward branching. The cost-duration analysis is treated as a minimum cost selection from explicitly enumerated arcs

selected according to a "Properly Oriented Cut Section" (POCS) procedure. The procedure is superior to tabular methods, but assumes linear or piece-wise linear and convex cost functions. The solution is not applicable to non-continuous functions.

Ignall and Shrage [57] explain the use of the branch and bound technique for a 3 machine 10 job scheduling problem. The explanation includes the reduction in spans by replacing nodes with dominating nodes which can significantly reduce the number of nodes that must be considered.

Pritsker, Miller and Zinkl [89] present a solution for the n products on m machines by a branch-and-bound search technique. The technique gives an optimum solution to problems where the jobs n must be processed in the same order on each machine. The best heuristic methods were compared in two cases using the technique. The procedure was shown to be accurate 17 out of 30 times and 8 out of 10 were within 5%. The first case was solved for n products with a late penalty and the second for total minimum time.

In 1968, Davis [28] introduced an exact enumerative technique which systematically formed all possible feasible network combinations with resource constraints. All schedules greater than a shorter schedule were successively eliminated using a shortest route approach. The procedure is complex and requires large amounts of computer time, but does consider the multiple resource requirements of a single job. The resulting algorithm will accommodate up to fifty jobs and five resources.

Davis and Heidorn [29] treat the constrained resource problem with a combination of heuristic and dynamic programming techniques. The Resource Scheduling Method developed by Shaffer, Ritter and Meyer [100] is first used to eliminate undesirable schedules and then dynamic programming obtains the optimum schedule from the reduced number of alternatives.

In 1968, Patton [83, 84] formulated the constrained resource problem as a mixed integer linear programming problem and develops a Nodal Sequence Pruning Test to solve the problem with two branch and bound algorithms which use selective enumeration. The model is not applicable to construction scheduling since the method for applying multiple resources is independent of time.

Other Techniques

Graph Theory

Graph theory and graph theory applications of the resource problem have been proposed by Berge [12], [13], Busacker [18], Dunne [35], Ellwein [37], Ford and Fulkerson [45], Gorenstein [51], Hu [55], Kaufmann [61] and Zangwill [114]. Generally the concept of a cut-set in a graph is defined as a set of arcs which disconnect a graph when the arcs are removed. The cut-set is useful for identifying limits of flow in graphs; however, the enumeration of all possible cut-sets is a sizeable problem for even moderate networks and consequently practical application is limited.

Based on the work of Roy and Sussman [96], Balas [4, 5], and

Raimond [93], Gorenstein [51] formulated the machine scheduling problem as a disjunctive graph. Disjunctive arc pairs were inserted into the network between nodes and one arc was selected to determine the derived network that represented the sequence of job processing on each machine. An algorithm to determine feasible networks was developed and partial enumeration used to solve the problem. Allowance for more than one resource to work on an activity and possible inclusion of a cost consideration was made.

More recent approaches by Dunne [35] and Dessouky and Dunne [30, 31] solve the cost duration problem for planar networks. The graph theory concept utilizes properly orientated cut-sets (POCS) which are a minimum set of arcs cutting the arcs of the planar network. The analysis is conducted on this dual type, cut network which actually forms a reduction set. The reduction set in an activity network is a minimum set of activities which reduce project duration by an amount equal to the amount that the reduction set has been reduced. This procedure is valid only for planar networks and assumes linear or piecewise linear and convex activity cost duration functions. The method exhibits potential for improvement and is more suitable to manual computation than most.

An earlier planar network solution of the critical resource problem using the dual of planar networks was proposed by Zimmerman and Shaffer [115]. As a starting point, the assumption of infinite critical resources was made and every possible cut set considered for scheduling. The method is called the "General Resource Scheduling" algorithm and is limited in application due to its dependence upon the basis of a planar network.

Nodal Balancing

Berman [14] developed an activity cost slope balancing technique to approach the CDA problem. Conceptually simple, the technique involves equating the sum of all activity cost slopes incident upon a node to the sum of the activity cost slopes emanating from a node. Balancing all nodes successively causes convergence to an optimum solution. While the approach has relative validity, Berman recognizes the limitation of the lack of consideration of manpower loading.

Taylor and Walsh [106] propose an interactive procedure where resources are categorized as to the effect that the application of the resources will have on the activity. Those with secondary effect are minimized first leaving only the primary resources to be assigned. The interactive method effectively allocates resources by parts until no additional resources are available.

Chapman [23] suggests the application of a transportation format with additional constraints to prevent non-feasible linkages where the float of one activity drives another activity into an infeasible region. The solution framework is first arranged similar to the standard transportation problem and an initial feasible solution generated. Tests for optimality are made in the standard transportation form. Subsequent interactions to gain an optimum solution are based on evaluation of row and column shadow costs. The problem of non-feasible corner points due to the introduction of additional constraints is encountered and the solution is proposed by simplex row evaluation techniques which effectively bridge non-feasible corner points. Chapman notes that optimal solutions should

be possible and additional effort to minimize computation is clearly necessary. In general, the adaptation is comprehensive and promises a positive contribution to the problem.

The methods presented depend upon computer solution of practical size problems. Non-computer, manual methods are unusual due to the complexity of even moderate size problems. Methods emphasizing a manual capability have been proposed by Fondahl [44], Antill and Woodhead [1], Martino [74], Waldron [110] and Siemans [101].

Research Needs

This review has examined a variety of approaches to essentially two problems: (a) Cost Duration Analysis and (b) Critical Resource Analysis. The methods available to solve these problems range from heuristics to programming (linear, integer, dynamic and quadratic) and combinatorial methods. Among these methods, only the programming techniques can guarantee optimality at the expense of severe limitations in scope and compromising assumptions.

The approaches examined almost exclusively treat the CRA and CDA problems separately and design solutions to solve either one or the other without recognizing the possible interaction. As noted by Paulson [86] time-cost trade-off methods generally ignore resource constraints and become engrossed in composite algorithms. Margenthaler [71] reported that several procedures have been designed to treat CDA and CRA alternately or in series, [2, 3, 24, 36, 72, 73 and 76]. However, only Dunne [35], Marganthaler [71], Paulson [86],

Richards [92] and Rosenbloom [94] state the need for a combined approach to the problem. Of those recognizing the interrelationship Marganthaler and Paulson present heuristic techniques and Richards proposes a highly complicated 0/1 method which is designed as a measure of effectiveness for heuristic techniques.

Research needs may be focused on obtaining a programming solution of the combined CRA and CDA problems without imposing assumptions such as linearity which remove the solution from applicability to the construction environment. However, even though mathematical rigor and precise assumptions are important:

> Any representation of an actual situation by a mathematical model is based on any number of compromises, aggregations of effects and approximations. If too much detail is included, application of the model becomes a prohibitive task. Moreover, the important relations may be almost completely obscured by the maze of detail. On the other hand, a model that does not include essential characteristics of the system has little utility. Thus it is highly desireable to have a model that has great flexibility for representing situations but whose application does not require an excessive amount of effort, Taylor and Walsh [106].

Such a model would have the capability of being used by the construction industry. As evidenced by the deficiencies of both the individual and composite efforts of the state-of-the-art this practical capability of generating an economical schedule without limiting assumptions has not been met.

CHAPTER III

DYNAMIC MODEL DEVELOPMENT

Introduction

The dynamic resource model is developed to study the alternative designs of the construction project planning and scheduling system. The information studied is the different resource requirements, the resources available to satisfy the requirements and the effect of the alternative resource decisions on the construction project.

The dynamic resource model considers the cost of alternative courses of action by evaluating the decisions in terms of the total project cost. Accordingly, all decisions within the system are interrelated to reflect upon the measure of effectiveness of the total project. The cost of each individual job then becomes a function of the minimum cost of the total system.

Total Project Costs

The cost of a construction project is the summation of the cost of all resource decisions, plus the cost of maintaining the project in an active status and the costs incurred as a result of liquidated damages. The resource decision costs result from the cost of using resources, the cost of having idle resources and the cost inherent in changing the level of resources.

Resource Utilization Costs

Construction project material is converted into a completed project component by applying the required resources over a discrete time period. Within practicable limits, which are controllable by an experienced scheduler, the discrete time required is inversely proportional to the resource applied in feasible discrete mixtures. For example, if the resource requirement is to transport 4,000 cubic yards of material and a feasible resource mixture of one loader and three trucks could move 1,000 cubic yards daily the time of resource application would be four days. If practicable, two loaders and six trucks could accomplish the requirement in two days.

The cost of using the resource is the total cost associated with employing the resource. In the example cited, the cost would include:

a. Depreciation cost of equipment

b. Interest, insurance and taxes of the equipment

c. Equipment operating costs

d. Total operator employment cost

The calculation of the sum of the costs in this example as well as with most resources is reasonably constant for a maximum of an eight-hour day. Correspondingly, the daily cost is constant for a maximum five-day work week. Beyond either the eight-hour day or the five-day work week, the resource costs will increase due to overtime and increased operating costs.

Resource Idle Costs

The cost of employing a resource and not applying the resource to a

project differs from the utilization cost. In the equipment example, the depreciation, operating and perhaps the operator's cost would change if the equipment remained in an idle state. Generally, personnel in the idle state cost the same as if they were applied. However, some union contracts allow payment of a minimum charge such as a four-hour day which would change the total employment wages and benefits considerably if the workman was sent home with a half workday credit. In other situations, the operator may be reassigned to another job with no cost to the idle resource.

Changing Resource Level Costs

Resource levels are the number of resources or resource mixtures which are able to independently accomplish work on a project component. A minimum level has an associated cost which reflects the least cost of having that resource mixture or resource on the site. If another unit of resource (or resource mixture) is applied, a cost that is not part of the unit resource cost is incurred. While there is the possibility that the cost is zero, most frequently a minimum of an administrative cost of ordering the resource and placing the resource on accounting records occurs.

Changing resource levels include costs such as the following:

a. Administrative Costs per Unit Resource
b. Erection/Dismantling of Resource
c. Transportation of Resources
d. Inefficient Learning or Break-in Period
e. Training Costs Associated with Resource Use

In general, any cost associated with a resource which does not represent a reasonably constant utilization cost is a cost of changing the resource level. As shown by Figure 14 the costs during periods t=0 to t=6 are not representative of the resource utilization and should be considered costs of changing level, regardless of their source.

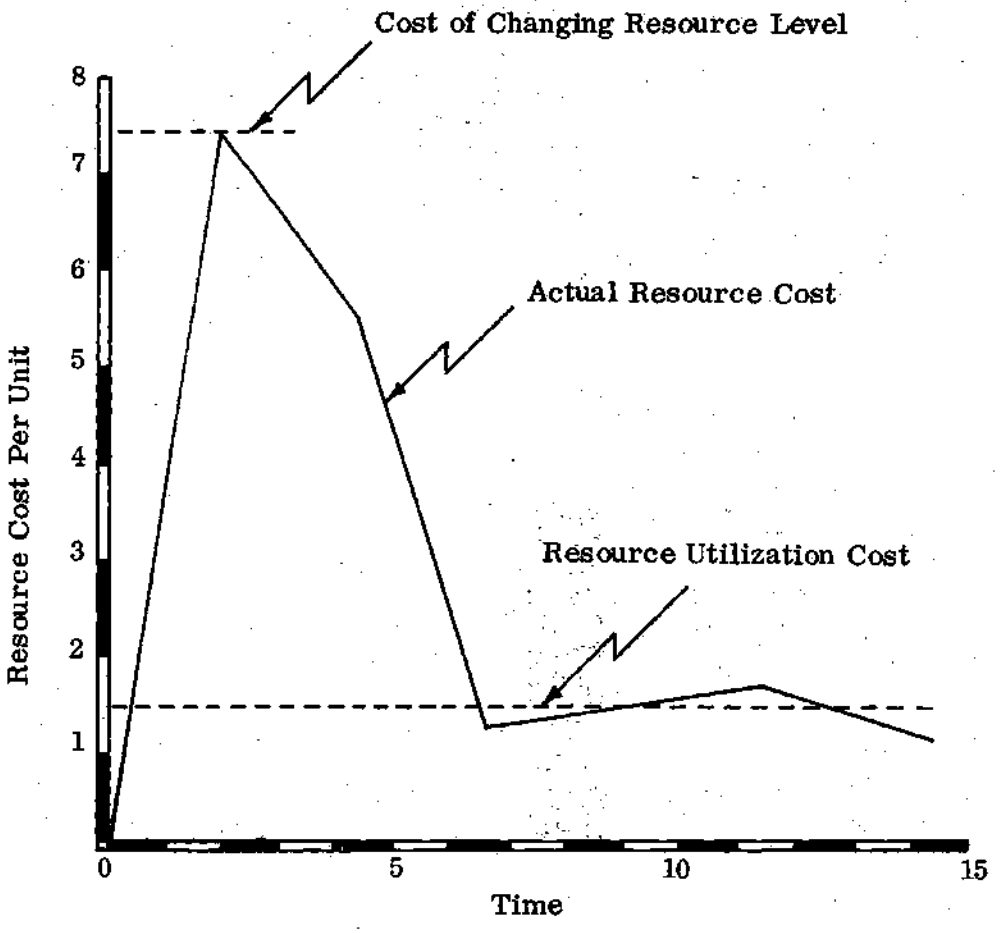


Figure 14. Resource Cost per Unit vs. Time of Application.

Active Project Costs

A construction project that is actively pursued requires support that includes some construction support resources and additional indirect job costs. These costs result from supporting the scope of the project and are required to assist in the actual creation of the project components. Following is a list of support costs which are included as Active Project Costs:

a. Home Office Overhead Prorated to Project

b. Project Site Officer

c. Indirect Labor - Superintendent

Assistant superintendents

Craft supervisors above crew leaders

Watchmen

Cleanup men

Inspectors

Timekeepers

Accountants

Schedulers

Expeditors

Storage supervisor

Safety engineer

Secretaries

Inventory clerks

d. Site Services - Water

Electricity

Temporary sanitary facilities

Dewatering/ drainage

Road patrols

e. Security Fencing

f. Construction Risk Insurance

g. Permits, Licenses, Fees

h. Telephone Services

i. Consulting Services

j. Computer Services

k. Warehousing (on and off site)

1. Storage (on and off site)

m. Prefabrication Area

n. Material Staging Areas

o. Materials Testing Services

p. Administrative Transportation

The applicable costs listed above constitute a carrying cost of running a project. The longer these costs apply the greater the total Active Project Costs will amount to. Consequently, a measure of reducing project duration by one day that costs less than the sum of the applicable Active Project Costs would be cost effective.

Delay Damages

In addition to the indirect costs of running a project, an extra cost of completing a project late may be incurred. Delay damages, if applicable, are effective after the project completion date stipulated in the contract specifications. Delay damages may also be applied to selected intermediate project components prior to the overall completion date. As with active project costs, delay damages exceeding their avoidance cost should not be willingly incurred.

Project Material Resources

To round out the total project costs, the cost of the project materials must be considered in conjunction with the processing costs. These costs are variable, but alternate decisions are severely limited by the specifications unless a reasonable value engineering clause is included. In any event, the project material costs will be considered fixed at the level priced from the material take-off and the possible value engineering decisions excluded from the scope of this research.

Project Cost Curves

The total project costs are represented in Figure 22 as a cost c and the project time at point t. If the resource level is not initially at a maximum or a minimum, then the project time can be reduced by some δ to a point c', t' where all critical jobs have a maximum resource assignment. Likewise, the resource level of critical jobs can be reduced until the lower resource limit is reached at c'', t''. If feasible completion times exist between c', t' and c'', t'',

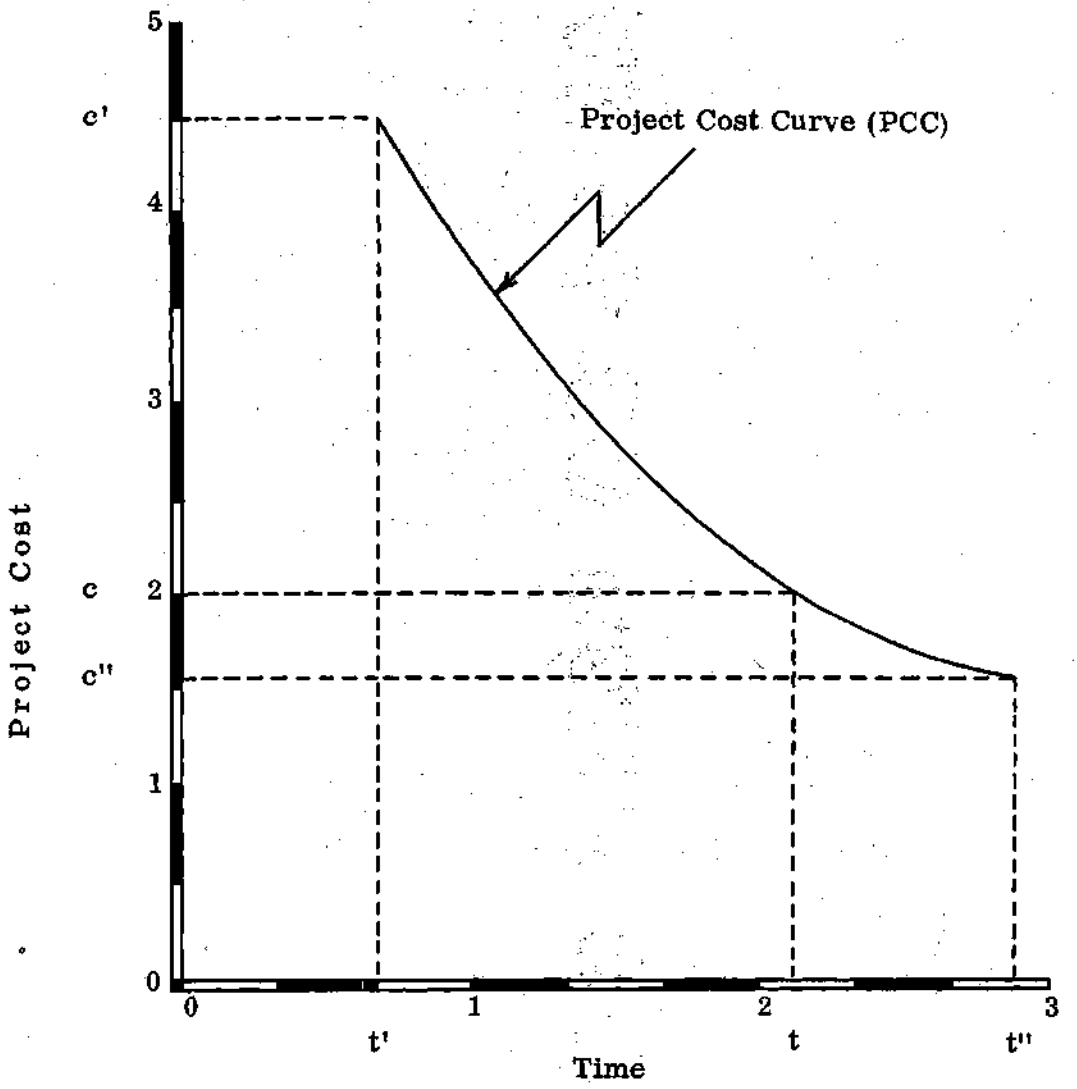


Figure 15. Example Project Cost Curve.

an apparent project cost curve (PCC) can be established. A curve such as shown by Figure 15 may represent the cost of project resources, idle costs and the cost of changing resource level. The PCC represents all of the resource costs as well as all of the project components in the project. As such, a PCC may be considered to represent a series of sub-projects as shown by Figure 16.

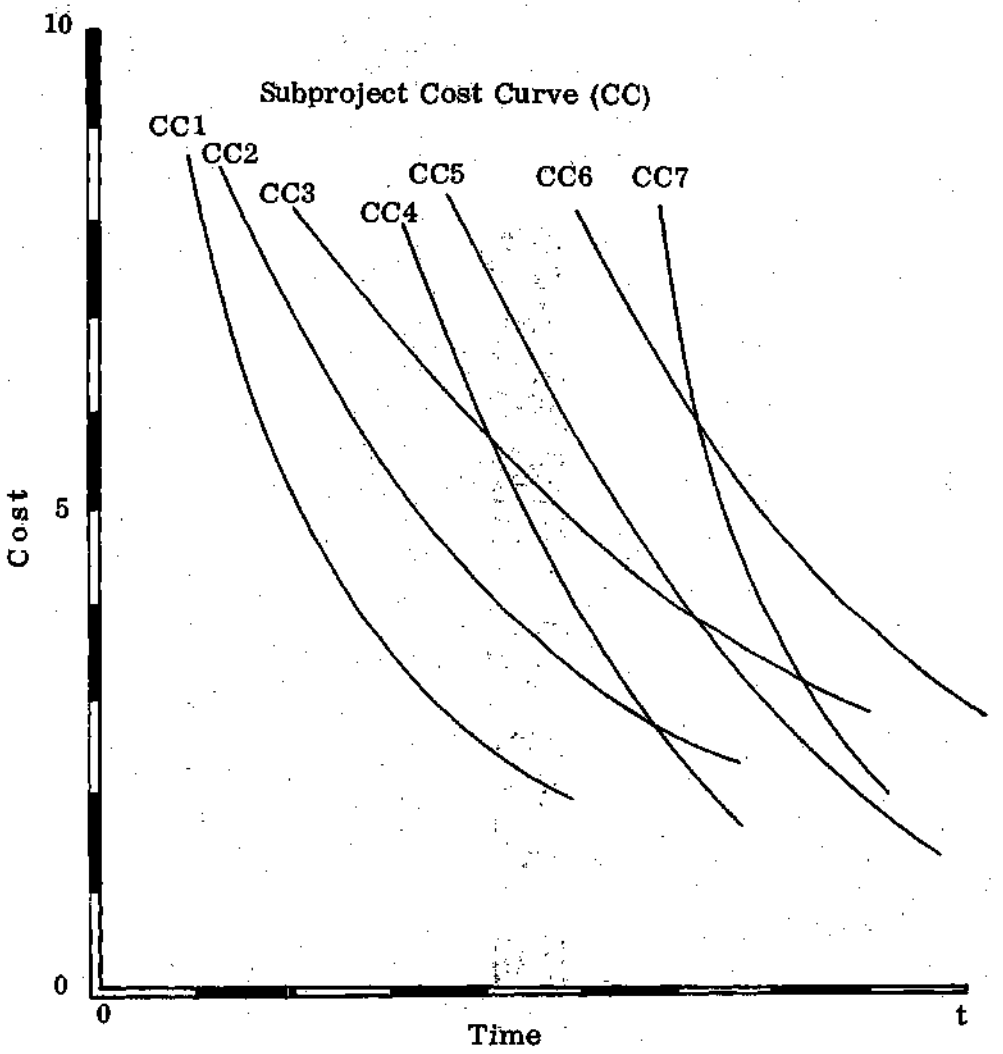


Figure 16. Cost Curves for Subprojects.

Cost Integration

The alternative decisions to apply resources are interrelated in three respects. First, the decision to apply a resource x at a level n affects the decision to apply a different resource, y, at a level m. Figure 17 shows two activites, both critical and incident at node k. The decision to apply 2 m of y resources may reduce the duration of j-k by some Δt_y at a cost of

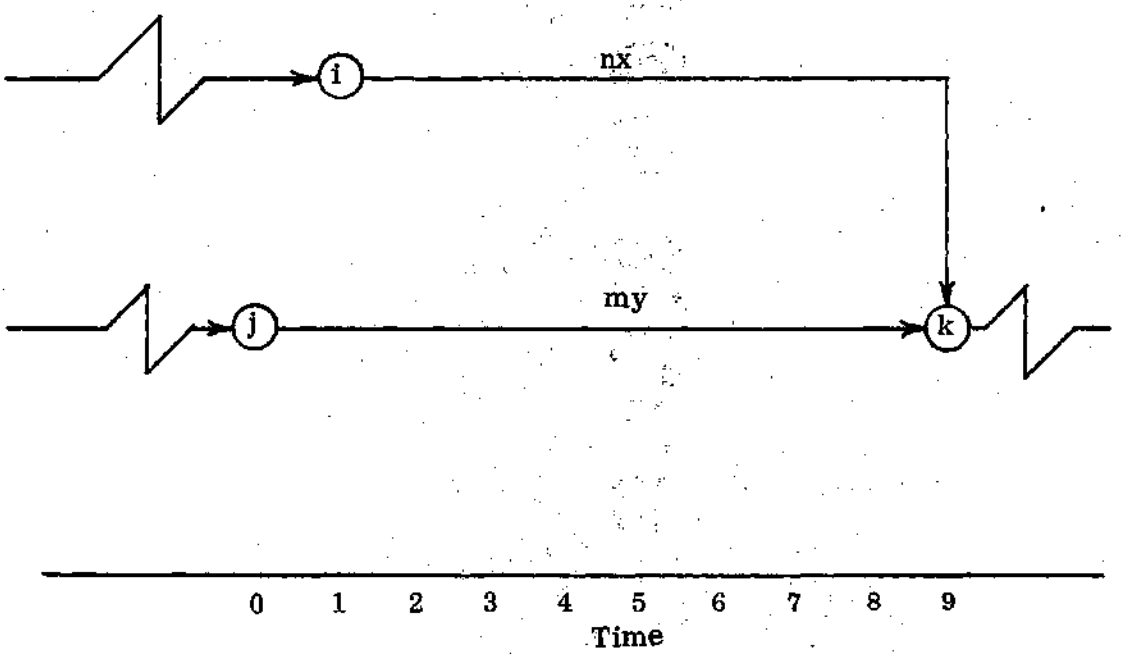


Figure 17. Time Scaled Activity Relationship.

C_{2m}. The project duration would remain unchanged unless i–k was also reduced by an amount $\Delta t_x \geq \Delta t_y$ at some cost C_n. Assuming that project durations less than one day are impractical, a Δt_x must be found that is $\geq \Delta t_y$. This reduction in duration may not be an optimum cost for completing i-k, j-k at a time $(t + 9) - \Delta t_y$. The costs of all feasible durations of i-k, j-k existing at a time t must be candidates for optimum costs. Regardless of eventual optimums, the decision regarding one resource affects the decision of other resources in the system at the same time.

The second dimension of cost integration is between discrete points in time. The opportunity to vary resource assignments within their respective limits exists from one discrete time to the next. The only incentive (or

deterrent) to change resource assignments is what it will cost not to make the change. Figure 18 shows three critical activities all requiring the same

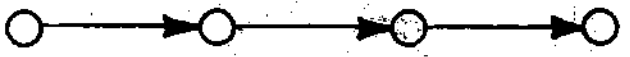


Figure 18. Critical Activities Requiring One Resource.

resource. If there are x_i feasible levels of resource x, where i=1, n and A is a requirement for resource x, $(A > x_n, < tx_n, t \in T$ feasible) there is a series of $\sum_{i=1}^{n} tx_i > A$, t=1, . . . , T and a series $\sum_{i=1}^{n} tx_i > B$, $\sum_{i=1}^{n} tx_i > C$. The cost of satisfying requirement A is entirely independent of satisfying B. However, the state of the system when A is completed is related to B since the level of the state when A is completed determines the amount of change necessary to transform the resource state to satisfy resource requirements at B.

The third dimension interrelating resource decisions is the cost of time. Figure 19 shows four resource requirements A, B, and C which are critical and D which is non-critical. If requirement A, B or C cannot be met in one time period, the critical path will be extended and the cost of that extension will accrue. Requirement D can be met concurrently with either A or B or can be accomplished in part with the work of A and part with B. This cannot happen without certain assumptions. First, if the work is not accomplished with A, some (in this case all) of the float will be used. Second, if resources

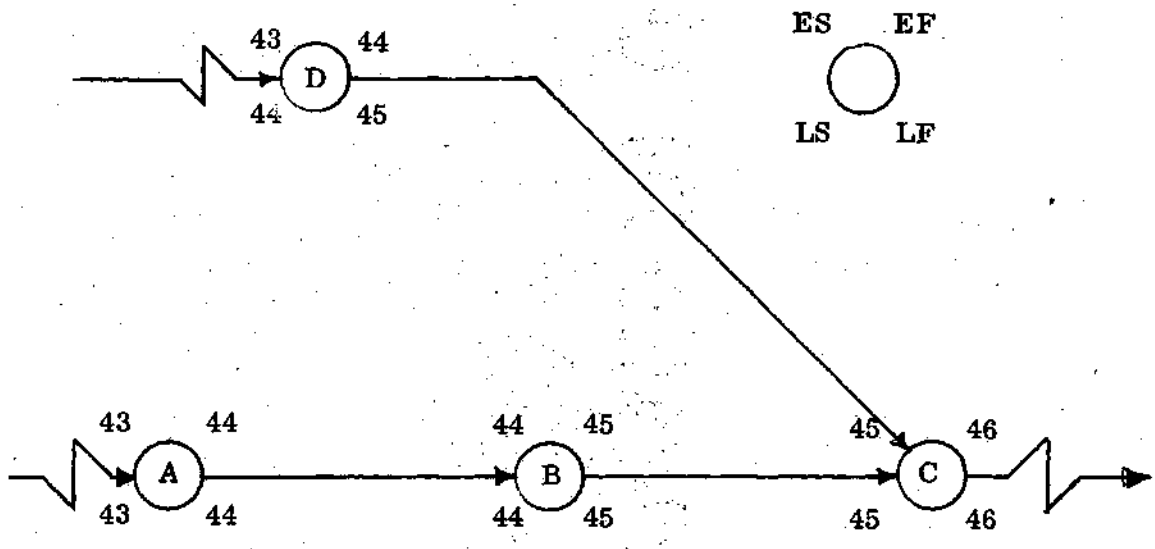


Figure 19. Critical and Non-critical Activities with One Resource.

are only available in mixtures, one complete resource mixture unit must be assigned beyond the requirements of A and B if any additional effort is available for D.

The assumptions inherent in this problem are then twofold:

a. Is the work separable into smaller packages which can be worked as resources are available?

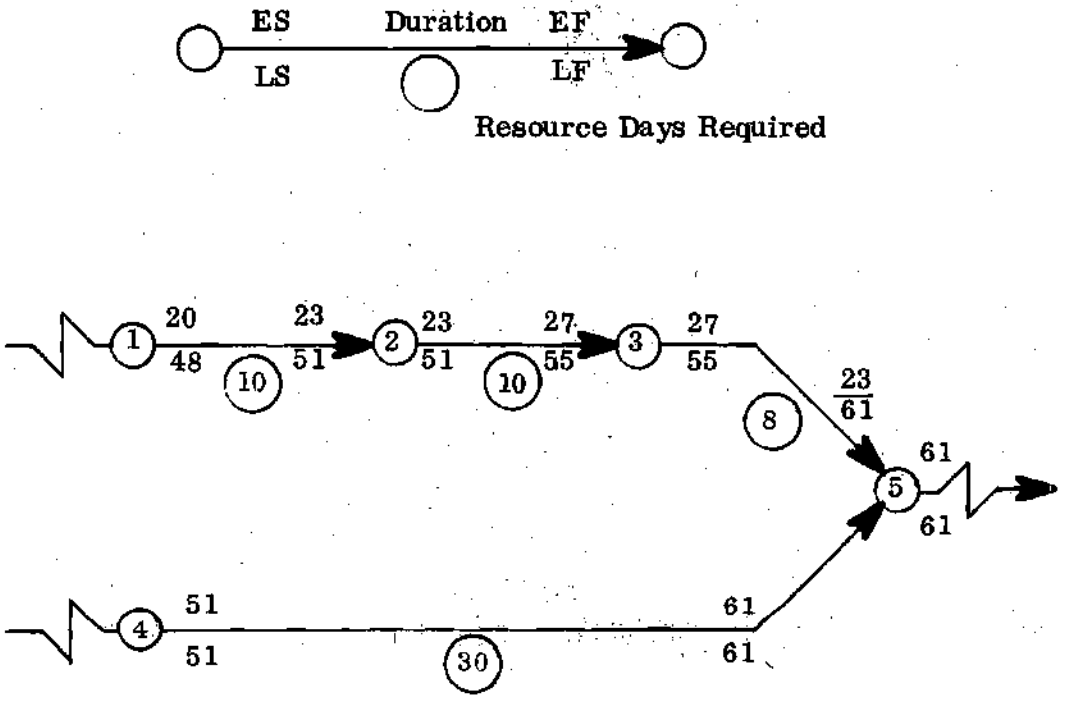
b. Will float be used or will the schedule be "left justified?"

In reference to the separability of jobs, Roy and Dibon [95] maintain that jobs should not be broken but remain continuous. Kelley [64] holds the opposite opinion that a large percentage of jobs are not required to be processed continuously. Neither position can claim universal application when the merit depends upon the individual job and the cost of stopping and starting that unique

operation. Unfortunately, imposing the task of determining the cost of separating jobs upon a scheduler is impracticable. Kelley's position that most jobs are separable is considered a reasonable compromise if the time of assignment of a resource is at least equal to one eight-hour shift. This position evolves from the reflection that some stopping and starting costs already exist by virtue of changing shifts or advancing to a future day.

The desirability of using float depends upon the willingness of the organization to accept an unknown element of risk. The risk results from exogenous variables which may cause a job originally having float to be delayed beyond its finish time. The alternatives of scheduling a non-critical job on its early start time and sacrificing the possible cost savings of using any float is approximately as undesirable as risking a delay due to scheduling an activity on the late start date. A compromise position can be established by defining a new concept of "partitioned float" which will represent a time to start an activity that is between the activity's early start and late start times. To determine "partitioned float," the total float available to a chain of activities is prorated on the basis of the resource days required by an activity as compared to the total resource days required by a chain of activities. This compromise is based on the argument that the more work required by a job the greater will be the opportunity for delay. Figure 20 shows this procedure.

The resource types having float assigned may be different. For example, activity 1 - 2 may require equipment while 2 - 3 and 3 - 5 may require cement finishers and masons respectively. Each separate requirement does exist in a



Activity	Resource Days Required	TF	$TF\left(\frac{R}{\Sigma R}\right)$	PF
1 - 2	10	28	10/40	7
2 - 3	20	28	20/40	14
3 - 5	10	28	10/40	7
4 - 5	30	0	0	0

Figure 20. Determining Partitioned Float.

non-critical chain which by definition has some amount of float that separates the criticality of the chain from a parallel, critical chain of activities.

Many non critical activities are so nearly critical that a practical distinction is not terribly useful to a superintendent. This is particularly true when the total float in a chain (in days) is less than the number of jobs in the chain. In such cases, it is only prudent to schedule all jobs on their early start. Other instances will exist where the float is greater than the number of jobs, but less than a multiple of the prorated job requirements. The remainder is more safely assigned to the last job in the chain.

Objectives of the Dynamic Model

Resulting from the resource costs and their integration, the alternatives which may produce the least cost is the desired end product. In this respect, the manager needs to know how much of a limited resource should be assigned to a job each day to result in a minimum cost for the total project. In addition to the required resource, the manager must also know what effect these decisions have upon the total project. Without this knowledge, a minimum cost that shows a project is currently above a previously established budget may be extremely difficult to justify. These considerations create a conflict between output use and report generation. First, the information is only reasonably accurate for the foreseeable future; a period as short as one working month. To generate scheduling information much beyond the time that exogenous effects can be predicted with some degree of accuracy is not only expensive, but has already been

noted to be of little actual utility. Consequently, scheduling information should be generated for a limited future which, although dependent upon specific situations, should not exceed approximately sixty working days. To remedy the lack of total project information, planning resource assignments can be carried to project conclusion utilizing data developed on a broader scope basis.

Scheduling Information

Scheduling information informs the construction superintendent of the specific limited resource level which should be assigned to a limited resource requirement during discrete scheduling periods. Subject to the cost of changing resource assignments, the superintendent can make changes as frequently as he may choose. However, a project of significant size would degenerate to chaos if resource assignments are made more frequently than on a daily basis. Additionally, resource assignments on a daily basis allow for a reasonable opportunity for economy resulting from changing resource levels while maintaining the opportunity to benefit from a continuously level assignment throughout the scheduling period.

The data used to generate scheduling information is provided by the scheduler and results from a detailed study of the resource requirements of the project components existing on the lowest level of the project work breakdown structure. These are the components which exhibit unit costs developed from quoted material prices and for which the production rate of men and equipment is reasonably well established. By limiting the time period for producing this detailed information to a maximum of sixty future working days,

the workload of the scheduler (scheduling team) is retained within a practicable realm of feasibility. The scheduling level is shown by Figure 21.

Planning Information

Information used for planning purposes can be of a more general nature. It is necessary to know far in advance that a critical resource will be required. The exact details of that requirement depend to a great extent upon the outcome of the exogenous influences which are operative during the ensuing time before the requirement has matured. Consequently, the data used to generate planning information results from the study of less detailed resource requirements. While organizational policy as well as capability of generating broad resource requirements differs greatly between organizations, the level of the work breakdown structure used to generate planning information should be keyed to two levels below that of a project containing significant structures.

Application to Construction Resources

The application of these considerations to the construction problem may take the form of a project requiring loaders, graders and trucks. The resource types would then be the different types of equipment required and the costs of operation including equipment operators would be included in the cost of the resource type. A typical resource requirement in this case would be estimated in machine hours to accomplish the required work. For example; three loader hours may constitute the loader resource requirement, the trucks may require five hours and the grader ten hours.

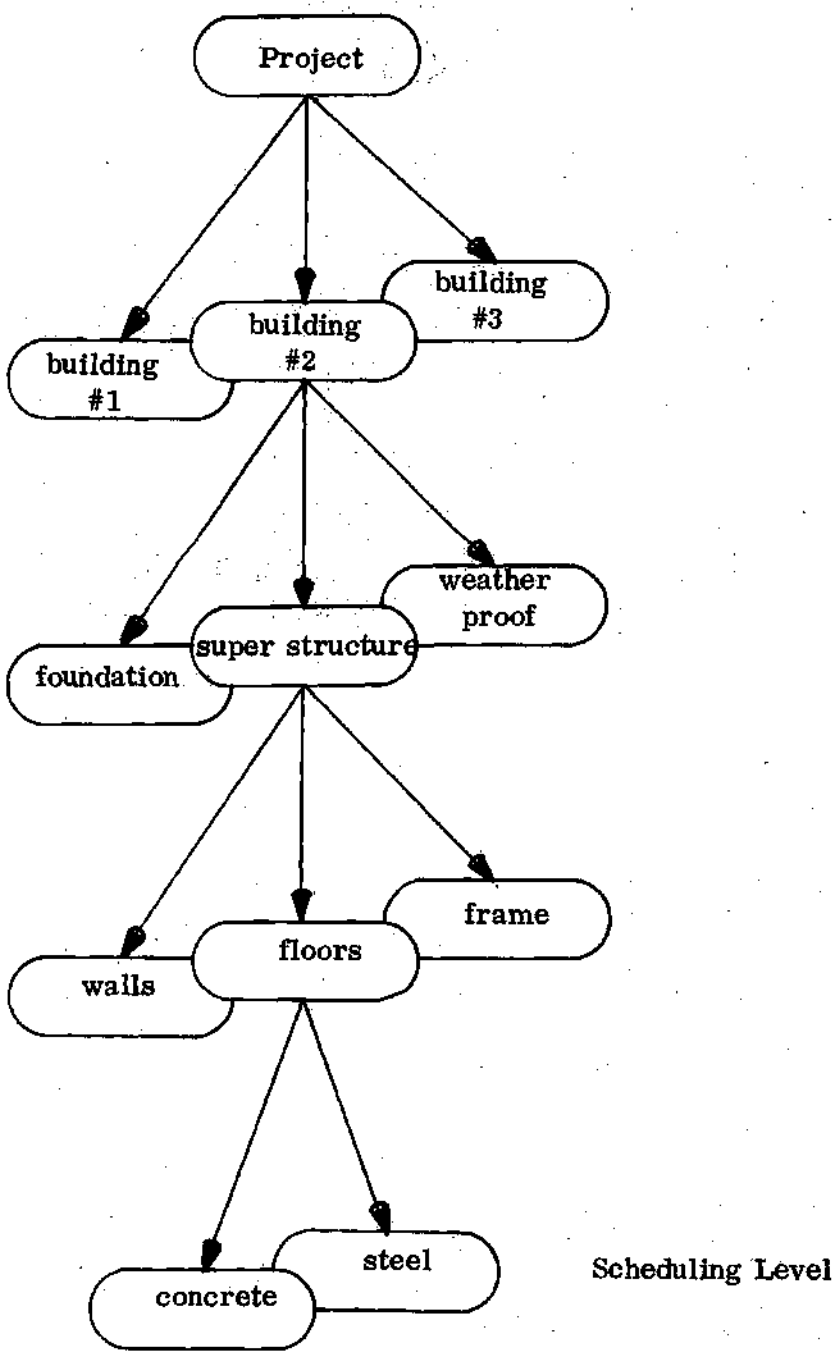


Figure 21. Work Breakdown Structure-Planning and Scheduling Levels.

The specific numbers of these resource types may be limited. There may be the alternatives of one to three loaders; six to nine trucks and four to seven graders. Additionally, there may be an active project cost which includes a project superintendent, a grade superintendent and home office overhead. In this situation, the superintendent knows the logic of his operation: He must load the trucks, haul the material and spread it on site. He may not know the number of each resource type which will result in the least cost when all the costs of the project are considered. If the problem is extended to a situation where more than one isolated part of the project must be accomplished simultaneously, the superintendent may also desire the information regarding how to apportion the limited resource types over his entire project to obtain the least project cost.

To find the lowest project cost, the superintendent must have the following information:

a. The cost of using each of his equipment types for a unit of time.

b. The cost of adding or releasing one unit of each equipment type.

c. The cost resulting from allowing one unit of each type of equipment to remain idle.

d. The active project cost.

The superintendent then needs to know how to combine these applicable costs to accomplish all of his requirements without spending more than is necessary.

This problem can be solved using dynamic programming. The solution process requires the adaptation of Discrete, Differential, Dyanmic Programming [53] to construction resource requirements; the provision of a new dynamic

programming method to allow simultaneous consideration of both critical and non-critical resource requirements (Binary Dynamic Programming); and a new method of releasing non-critical resource requirements to approach criticality (Partitioned Float).

Summary

The cost of a total construction project results from the following elements:

a. Resource Utilization Costs

b. Resource Idle Costs

c. Costs of Changing Resource Levels

d. Costs that are a function of project time

Each of these costs are interrelated between resource types and as a function of time. The objective of manipulating these costs is to minimize total project cost. The partitioned float concept allows use of a proportional amount of total float belonging to a chain of non-critical activities and is related to the "critical sequence" concepts of Weist [113] and Paulson [85, 86].

The resource scheduling problem and the resource planning problem have the same objective. Scheduling information must be detailed, but is only useful for a short foreseeable, scheduling horizon. Planning information is much less accurate and results from resource estimates made on a broader basis than that of scheduling information.

Both planning and scheduling information is necessary to carry out a minimum cost, resource management decision.

CHAPTER IV

MODEL FORMULATION

The dynamic resource model is formulated with an objective function and a set of constraint equations for each project. A project is defined by an arrangement of jobs in a directed, acyclic network. Each job of the project is characterized by resource utilization costs which are a function of job time. The objective of the model is to minimize total project cost through optimum resource assignment.

General Formulation of Dynamic Programming Problems

Bellman [9] introduced dynamic programming (DP) as a method based on a principle of optimality and designed to solve certain classes of problems. Problems suitable for DP solution are those which allow decomposition into a series of sequential problems of less complexity. The resultant series of solutions are then combined to obtain the solution of the original problem. Consequently, problems which require the solution of a sequence of interrelated decisions are conveniently formulated as DP problems. Conveniently, the construction resource scheduling problem consists of the decision to assign resources during a series of discrete time intervals and is well suited to the DP method. Problems suitable for DP analysis should also exhibit the following characteristics:

a. The problem must be capable of being divided into stages with a decision required at each stage. The DP methodology converts an optimization problem into a sequence of N smaller problems, each of which has a value of all N variables at a point in time.

The selection of the stages for decomposition may be points in space or time, or may even represent abstract steps in the problem-solving process. In construction scheduling analysis, optimization problems arise in the time-oriented optimization of the system. Stages may be widely separated points or very small increments of time. The stages of a construction resource problem can be conveniently specified as one day (eight hour) increments for scheduling.

b. Regardless of the type of stage, each must have an associated state vector. A state vector is a set of state variables which contain all the practicable information of interest relating to the state of a system at the associated stage. The information about the state variables must show how the states vary between stages. Multidimensional state vectors exist when two or more variables are necessary to describe the system's state. Realistic formulation of optimization problems in construction systems involves a high-dimensional state vector. The state variables applicable to the construction problem are the types of resources required within a stage or the resource mixtures required within a stage.

c. A transformation equation at each stage is used to transform the current state vector into a state vector which is associated with the next stage. Consequently, the decision vector is a set of variables representing the alternative decisions which may be made at each stage. The construction decision

involves the quantity of a resource requirement which will be accomplished within each stage (i.e., assigning work to the resources available). The result of these alternative decisions is then evaluated by a measure of benefit for any feasible state of the system at each stage. The characteristic of a measure of benefit must reflect the objectives of the system. In this respect, the objective of the construction system is to minimize the total project cost which is equivalent to the sum of the costs (benefits) of each stage).

d. For a specific stage and state of a problem, the optimal decision should be independent of decisions made in previous stages. This means that the information about previous stages which relate to the selection of new optimal values of decision variables is already incorporated into the values of the state variables at the current stage.

The recursive equation is the basis for the formulation of optimization problems which will be solved by dynamic programming. The recursive equation states mathematically Bellman's principle of optimality: "an optimal set of decisions has the property that whatever the first decision is, the remaining decisions must be optimal with respect to the outcome which results from the first decision."

When an optimization problem can be divided analytically into N discrete stages and each stage n(n=1, 2, ..., N with n=1 for the first stage, etc.) has an associated state vector $\bar{S}_n x$, then $T_n( )$ can be defined as a transformation function which acts on the state vector $\bar{S}_n$ to convert it into a state vector $\bar{S}_{n-1}$ (associated with stage n-1) which results from the action of the decision vector

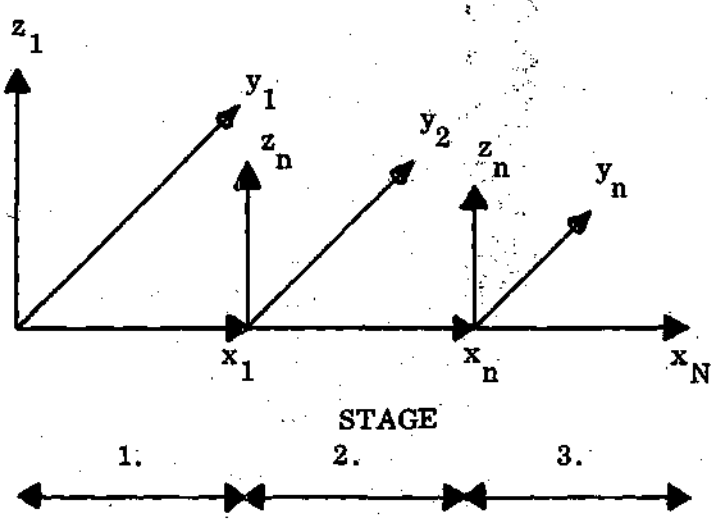
$\bar{D}_{n-1}$ in the (n-1)th stage. In mathematical terms and as stated by references [9 and 53]

$$\bar{S}_{n-1} = T_n(\bar{S}_n, \bar{D}_{n-1}); n = 2, \ldots, N \quad (4)$$

where $\bar{S}_n \epsilon \{\bar{S}\}_n$, $\bar{S}_{n-1} \epsilon \{\bar{S}\}_{n-1}$, and $\bar{D}_{n-1} \epsilon \{\bar{D}\}_{n-1}$; $\{\bar{S}\}_n$, $\{\bar{S}\}_{n-1}$ and $\{\bar{D}\}_{n-1}$ are the respective admissible values of the state vector for stage n, the state vector for stage n-1, and the decision vector for stage n-1. The construction transformation involves the amount of resource days required, to complete the part of the job not yet accomplished. The constrained or limited resource problem can be viewed in three dimensions as shown by Figure 22. An original stage time (t) is set equal to one day. Because a quantity of resource (y) which is available on day (t) may not be equal to the requirements for resource (y) at time (t), some additional time (Δt) may be required to satisfy the work requirement. Since the resource type requirement (y_t) is multidimentional (Y_{mt}), the additional time required to complete each resource type requirement may also be variable as shown by Figure 23.

Dynamic problems may be solved using either a forward or backward algorithm. Using a forward DP algorithm, the computation begins with the analysis of the first stage (n=1) and continues sequentially until the last stage (n=N) is completed. The backward algorithm begins with the analysis of the last stage (n=1) and continues sequentially until the first stage (n=N) is performed.

The measure of benefit or return resulting from the decision for stage n



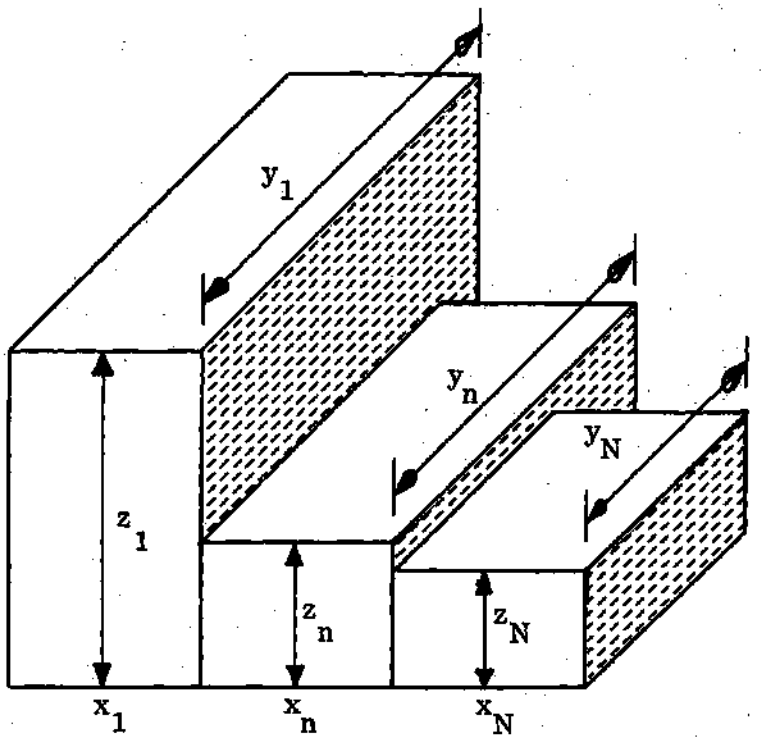


Figure 22. Three Dimensional Resource Synthesis.

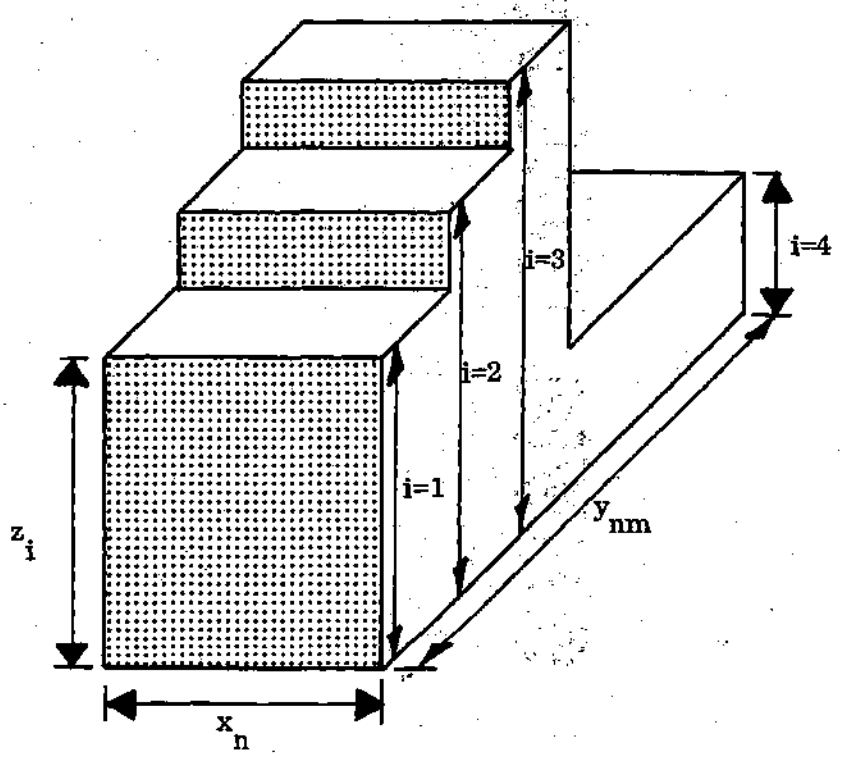


Figure 23. Multidimensional Resource Type Requirement.

is defined as $R_n(\bar{S}_n, \bar{D}_{n-1})$. The problem's objective is to minimize a given function for the benefits resulting from the sequence of decisions. The objective function, z, can be expressed as:

$$z = \text{minimize } f(R_1(\bar{S}_1, \bar{D}_0)), \ \ldots, \ R_n(\bar{S}_n, \bar{D}_{n-1}), \ \ldots, \ R_N(\bar{S}_N, \bar{D}_{N-1}) \qquad (5)$$

where f is a function representing the combined benefit of system transformations from the initial state vector $\bar{S}_0$ to a final state vector $\bar{S}_N$ which resulted from a sequence of decision vectors $\bar{D}_0, \ \ldots, \ \bar{D}_{N-1}$.

The optimization of the objective function is generally subject to a set of constraints or conditions of diverse nature (financial, physical, social, or

even institutional) which may be imposed upon the system. For example, physical constraints may specify limits to the allowable variation of the decision variables, cash flow constraints may indicate ceilings to expenditures necessarily associated with the decisions, and social or institutional constraints may restrict the system to the real-world situations where labor union contracts, local laws and company policy may impact the system. These alternative constraints applicable to the construction process were discussed in chapters I and III.

The solution of the objective function by DP requires the stage by stage solution of an N-stage problem. As noted the decomposition of $\bar{D}_n$ into stages requires two conditions. First, $\bar{D}_n$ must be separable as expressed by:

$$f[R_1(\bar{S}_1, \bar{D}_0),\ R_2(\bar{S}_2, \bar{D}_1),\ \ldots,\ R_N(\bar{S}_N, \bar{D}_{N-1})] =$$

$$f_1\ \{R_1(\bar{S}_1, \bar{D}_0),\ f_2[R_2(S_2, D_1),\ \ldots,\ R_T(S_N, D_{N-1})]\} \tag{6}$$

where f_1 and f_2 are real-valued functions. Secondly, $\bar{D}_n$ must be monotonic to allow decomposition as expressed by:

$$f\left[R_1(\bar{S}_1, \bar{D}_0),\ f_2'[R_2(\bar{S}_2, \bar{D}_1), \ldots, R_T(\bar{S}_N, \bar{D}_{N-1})]\right] \geq$$

$$f_1\ \{R_1(\bar{S}_1, \bar{D}_0),\ f_2''[R_2(\bar{S}_2, \bar{D}_1),\ \ldots,\ R_N(\bar{S}_N, \bar{D}_{N-1})]\} \tag{7}$$

where $f_2'[\ ] \geq f''[\ ]$ for all values of $R_1(\bar{S}_1, \bar{D}_0)$. Therefore $\bar{D}_n$ as stated by,

$$\text{Minimize } f[R_1(\bar{S}_1, \bar{D}_0),\ R_2(\bar{S}_2, \bar{D}_1),\ \ldots,\ R_N(\bar{S}_N, \bar{D}_{N-1})] \tag{8}$$

can be converted into

$$\text{Minimize } f_1\ \{R_1(\bar{S}_1(\bar{D}_0),\ \ \text{Minimize } f_2[R_2(\bar{S}_2, \bar{D}_1),\ \ldots,\ R_N(\bar{S}_N, \bar{D}_{N-1})]\} \tag{9}$$

Additive functions are typical in the analysis of systems which are designed to minimize the sum of the costs or the effectiveness of the system transformation at all the stages. In these cases, the objective function is given by

$$f[R_1(\bar{S}_1, \bar{D}_0),\ \ldots,\ R_N(\bar{S}_N, \bar{D}_{N-1})] = \sum_{n=1}^{N} R_n(\bar{S}_n, \bar{D}_{n-1}) \tag{10}$$

and when the conditions of separability and monotonicity are satisfied, f( ) is decomposable. In the construction system, the decision $\bar{D}_n$ is known to be separable by comparison to actual practice where the resource applied may be changed subject to resource constraints at any discrete time interval. Further, the decision $\bar{D}_n$ at any n is monotonic by definition as was expressed by the requirement that the number of feasible, minimum resource levels can only

increase with an increase in the number of resources. The monotonic feature is illustrated graphically by Figure 24.

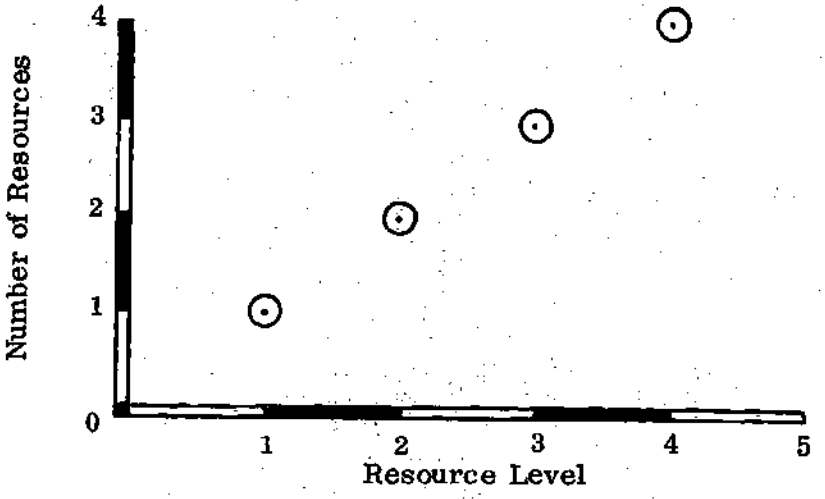


Figure 24. Number of Resources vs Resource Level.

If $F_N(\bar{S}_N)$ is defined as the minimum cost of the system transformations from the initial state $\bar{S}_0$ to the final state $\bar{S}_N$. Then,

$$\text{If } F_N(\bar{S}_N) = \text{Min} \sum_{n=1}^{n=N} R_n(\bar{S}_n, \bar{D}_{n-1}) \tag{11}$$

subject to the transformation function and all of the constraints imposed on the system. For any state (n(n=1, 2, . . . , N), the DP recursive equation where $F_0(\bar{S}_0)$ is a known quantity associated with the initial state vector $\bar{S}_0$ which represents the original condition of the system.

Computer Memory Required to Solve the DP Recursive Equation

The digital computer solution of the DP recursive equation, (reference 53) requires enough storage capacity to retain $F_{n-1}(\bar{S}_{n-1})$ and $F_n(\bar{S}_n)$, for all feasible

values of the state vectors $\bar{S}_{n-1}$ and $\bar{S}_n$ at two consecutive stages, n-1 and n, and the values $\bar{D}^*_{n-1}(\bar{S}_n)$ of the decision vector $\bar{D}_{n-1}$ which satisfies the requirement for every feasible value of the state vector $\bar{S}_n$, n=1, 2, . . . , N. This memory requirement may be large, as illustrated in the following example:

A typical case in the analysis of the monthly operation of a four resource construction system on a daily basis for three months involves N=60 stages, with a five-dimensional (M=5) state vector, and a five-dimensional (T=5) decision vector. In addition, if each variable of the state vector is broken into Q=5 levels, then, the computer's memory requirement is, at least, equal to $2Q^M$ for $F_n(\bar{S}_n)$ and $F_{n-1}(\bar{S}_{n-1})$ and equal to TNQ^M for $\bar{D}^*_{n-1}(\bar{S}_n)$, n=1, 2, . . . , N. For this example, the total numbers for $F_n(\bar{S}_n)$, $F_{n-1}(\bar{S}_{n-1})$, and $\bar{D}^*_{n-1}(\bar{S}_n)$ are 943,750. Storing these numbers may require approximately four million bytes, a quantity which is significant for the memory capacity of available digital computers.

It can be seen from the preceding example that the computer memory required for the solution of the recursive equation increases linearly with the dimensionality of the decision vector and the number of stages, geometrically with the number of values of the state variables, and exponentially with the dimensionality of the state vector. The most critical factor in the determination of the required computer memory is obviously the dimensionality of the state vector which affects the memory exponentially. However, for a given problem, the dimensionality of the state and decision vectors and the number of stages are fixed by the appropriate formulation of the problem, and the accuracy of the solution depends on the fineness of the division of the state and decision variables

into quantified values. There exists then, a compromise between the accuracy of a solution and the computational effort of obtaining it. For a minimum desirable number of quantified values of the state variables, the DP solution of a particular problem may not be possible if the entire set of values of the state variables is considered at one time. In such cases, a method is required which will allow a computer solution of DP problems having high-dimensional state and decision vectors, without requiring any reduction in the number of values of state and decision variables. These storage requirements are similar and are stated similarly to [53].

Reduction Technique

As a result of the excessive storage requirements created by a dynamic programming formulation, a method of reducing the dimensionality is necessary to allow solution by present computer equipment. Discrete differential dynamic programming (DDDP) as explained by Heidari, Chow and Meredith [53] is a constrained recursive search method which can be employed to solve optimization problems formulated in terms of dynamic programming (DP). The method requires an amount of computer time and memory that is relatively small when compared to conventional DP requirements. DDDP is an iterative technique that uses the recursive equation of dynamic programming to find an improved solution by constraining the search for improved solutions to domains which are successively closer to that of each solution. Employing this regimen, DDDP takes advantage of the results of each previous trial solution to adapt the search for

each successive, improved solution.

DDDP methodology is a solution technique for DP and consequently requires that the problem be capable of formulation as a DP routine. As such DDDP is not an aid to the formulation of a specific problem but a computational tool to obtain solution to an optimization problem.

The collection of restricted values of the state variables at all stages defines a corridor. The composition of corridors varies from one iteration to the next in such a way as to obtain convergence of the algorithm toward the optimal solution for the entire set of quantified values of the state variables. The path of the iterations through a corridor is called a trajectory (k).

A trajectory, either initial or optimal, is required for the formation of the corridor for each iteration. Associated with such a trajectory is a value $F^*_{(k-1)}$ of the objective function which is either calculated from the transformations of the system as specified by the initial trajectory (if k = 1) or obtained from the solution of the recursive equation within the corridor for the (k - 1)-th iteration (if $2 \leq k \leq k_{max}$, where k_{max} is a given maximum number of iterations). The solution of the recursive equation within the corridor yields a value of F* for the objective function. If the value converges, the trajectory represents a solution to the optimization problem. This procedure is the same as was used and proven by Heidari, Chow and Meredith [53] with a minor exception in the manner of choosing the successive values of state variables.

DDDP Procedure

The DDDP procedure is shown by the block diagram in Figure 25. This diagram shows the steps necessary to implement the DDDP. The system is composed of a series of cycles which are further broken into a number of iterations for each cycle. Each cycle starts with a trial trajectory and searches for an optimal trajectory at each stage. A cycle is complete when the search has converged to an optimal trajectory as specified by a criterion. The optimal trajectory within a specified category and the corresponding return are determined by DP methodology for each iteration. The number of cycles as well as the maximum number of iterations of each cycle are preset for economy of processing. The procedure terminates when either the convergence criterion or the maximum number of iterations is satisfied. If convergence is realized, the optimal solution of the problem is the optimal solution of the last cycle. Termination resulting from exceeding the allowable number of iterations for a cycle produces a near optimal solution and may require additional iterations or modification of the convergence criterion.

The proximity of the initial trial trajectory to the optimum determines the processing difficulty of the procedure. While a trial trajectory may be found by applying engineering judgement or an approximate method such as system decomposition, a special subroutine provides a list of feasible alternatives from which initial data may be selected.

Having selected an initial trial trajectory the values of the state variable are then limited to a specified corridor around the trial trajectory. The corridor

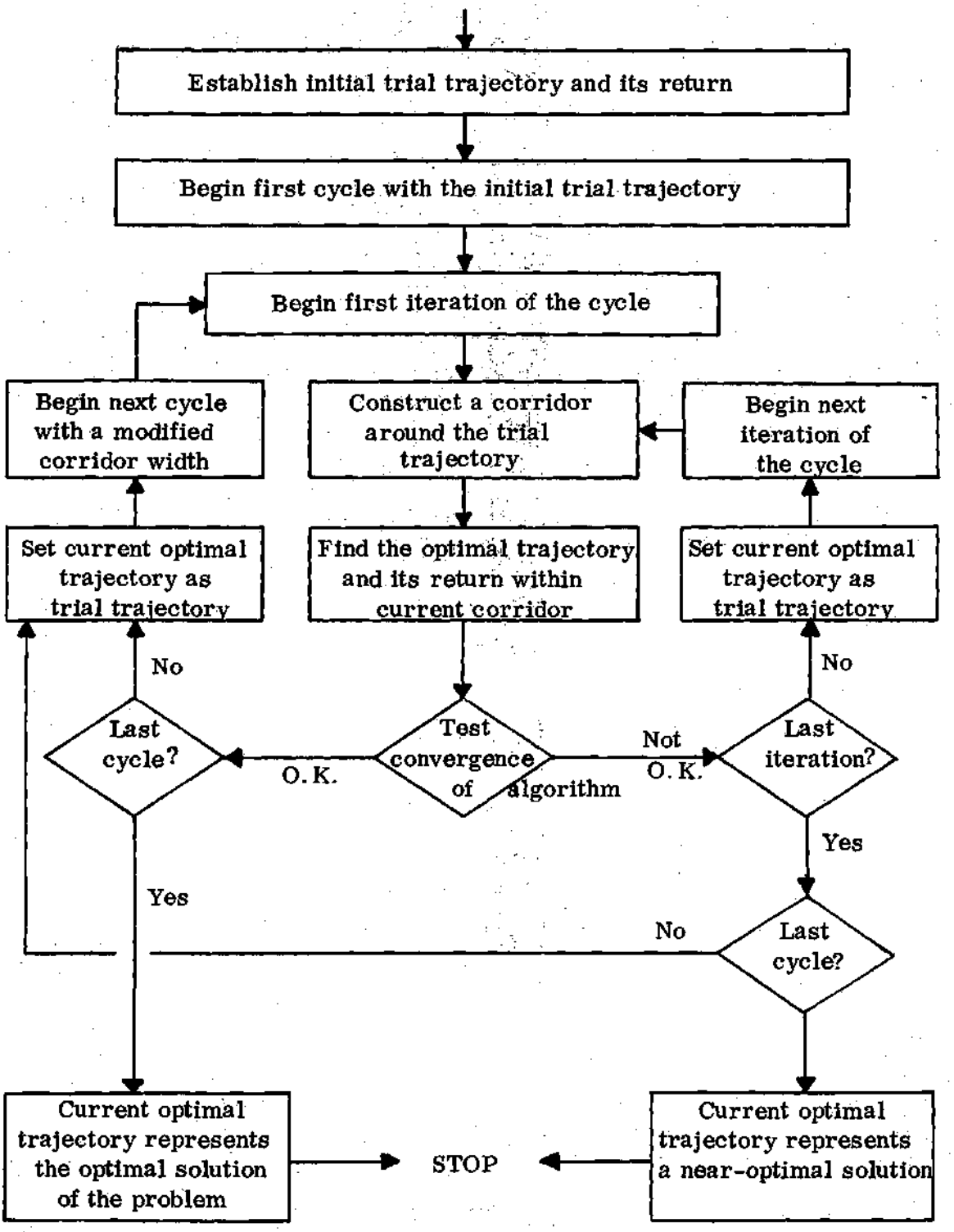


Figure 25. DDDP Procedure.

limits the values of the state variable which will be used for an individual cycle. The number of values that will be considered at one time should be reduced to as small a value as possible. Heidari, Chow and Meredith [53]; Shaufelberger [99]; Cortes [25] and Tauxe, Hall and Yeh [105] have shown the most satisfactory number to be three.

Figure 26 illustrates the concept of a three-valued corridor extending over N stages. The corridor should be constructed symmetrically around the trial trajectory if the maximum and minimum variable limits allow. When the corridor exceeds the range of feasibility for the variable, an asymmetrical corridor will result. Additionally, should the trial trajectory be incident upon either the upper or lower limit of the variable, a two valued corridor will result.

The width of the corridor $\Delta_{qm,n}$ should be the same for all stages during one cycle. However, the width may change between variables. Chow sets the increment $\Delta_{qm,n}$ equal to three times the smallest increment of the state variable. In the construction problem, the limited number of values which the state variable may assume (five*) allows simplification in determining the corridor width. Because the values of the state variable are so limited, setting the corridor width at a maximum and minimum value around initial trajectory will allow convergence in a reasonable, if not minimal number of iterations.

With the corridor constructed around the trial trajectory a conventional

*The five value state variable was set as a limit of scheduler capability. A larger range of values would require increased computation time, but would not hamper computational practability.

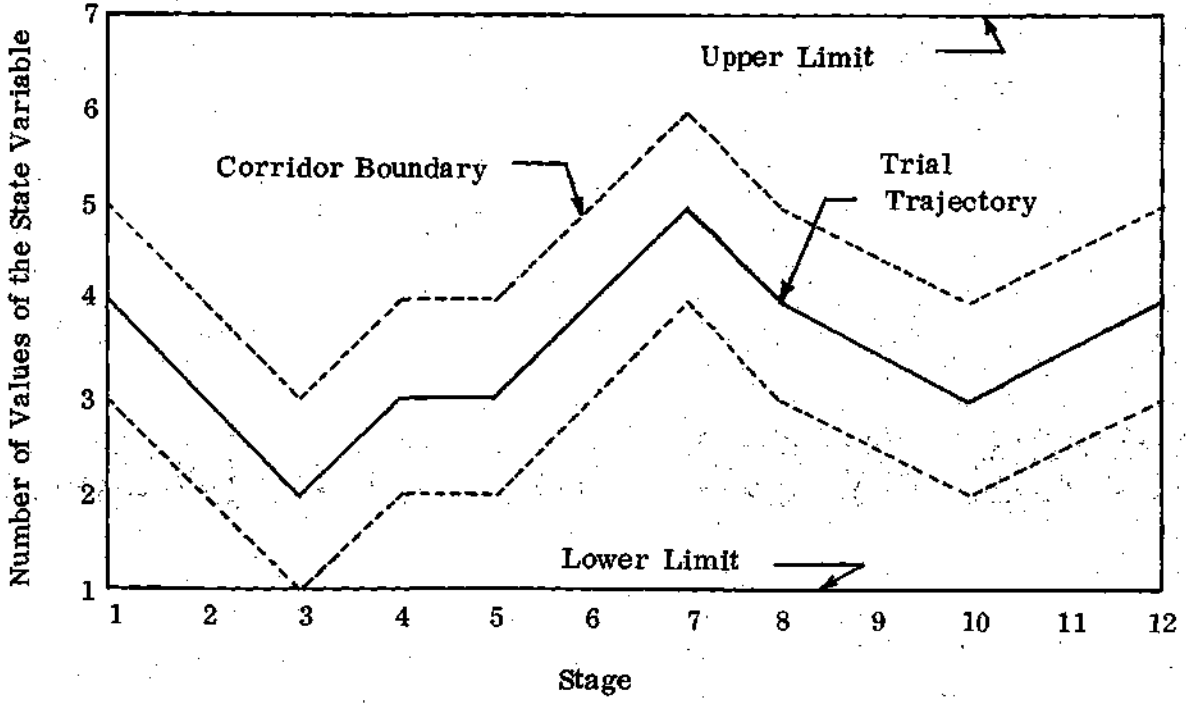


Figure 26. Construction of a 3-valued Corridor.

DP algorithm is used to search for the optimum of the values of the state variables included within the corridor. The value of the objective function can be obtained by either a backward or forward algorithm. The optimum trajectory can be retrieved by computation or traceback procedure in a direction opposite to that used for the DP recursive equation.

Tests for Convergence of Intermediate Cycles

This procedure is performed for each iteration of a cycle and each iteration is tested for convergence to the optimal trajectory. Each iteration further provides a trajectory with a return which is at least equal to that of the previous iteration. A measure of improvement of the returns from consecutive iterations, as compared to the first iteration is given by

$$\delta_i = \frac{F_i^* - F_{i-1}^*}{F_1^* - F_0^*} , \quad i = 1, 2, \ldots, I \tag{12}$$

I is maximum number of iterations per cycle

F_0^* is the return from the initial trajectory.

F_1^* is the return from the 1st iteration.

If, for any intermediate cycle the iterative process produces a $\delta_i \leq$ some value ϵ the process should be terminated. Chow recommends a value of ϵ equal to 0.10. Such a value while not insuring an optimum for an intermediate cycle does provide a reasonable trial trajectory for the succeeding cycle.

Test of Convergence of Final Cycle

The test for convergence of the last cycle consists of testing for a value of improvement which meets or exceeds a minimum requirement λ

$$\frac{F_i^* - F_{i-1}^*}{F_{i-1}^*} \leq \lambda, \; i = 1, 2, \ldots, I \tag{13}$$

For practical purposes, λ must be established in relation to the value of a project and may differ greatly from the λ used for an example problem. λ differs from ϵ in that λ should be much more restrictive for the last cycle. Chow suggests a $\lambda = 0.001$ which is within reason for the construction problem.

Project Parameters

A project P has a number of jobs N which must be completed using RT resource (mixture) types. Each job has a single resource requirement RR which represents the resource (mixture) days required to complete the job. The resources (mixture) have RL feasible levels each associated with the following parameters:

COST[RT, RL] = utilization of resource RT at level RL

IDLE[RT, RL] = idle cost of resource RT at level RL

IDC[RT, RL] = cost of changing level of resource RT from RL to RL+1

DLC[RT, RL] = cost of changing level of resource RT from RL to RL-1

The project P has the following parameters.

PCC[N, P] = active project cost incurred by extending job N one day on the critical path

DDC[P] = delay damage cost incurred by extending project P one day beyond the specified completion date

A project network has N activities numbered from starting node I to finish node J which specify the precedence relationships among the jobs. Associated with each job is an amount of total float and an amount of partitioned float where;

TF[N, P] = total float of project P, job N in days

PF[N, P] = partitioned float of project P, job N in days and

$$PF[N, P] = TF[N, P] \quad \frac{RR[N, P]}{\sum_{N-1}^{N} RR[N, P],\ N \in C} \tag{14}$$

where C is the set of the non-critical chain of jobs containing N and RR is the resource days required by job N, project P.

A project is planned for the entire duration T and scheduled for a short range future $\overset{t}{N}$. The discrete scheduling internal n is equal to one work day.

Constraints

The constraints required are minimal due to the requirement that input data is limited to feasible values. Consequently, job completion constraints are of primary importance. Each job must be completed within its own active range to maintain the required precedence relationships.

$$RR[RT, N] \leq \gamma[RT]\hat{n} \tag{15}$$

where γ is the work assigned in a time $\hat{n}$.

In any time period $\hat{n}$ the resource available must be greater than or equal to the resource assigned.

$$RL[RT] \geq \Sigma\ \gamma\ [RT, \hat{n}] \tag{16}$$

Variables

Each job N has a single resource-day requirement (RR[RT, N] which must be accomplished to complete job N. RL is a variable level resource (mixture) available to accomplish N. $\hat{n}$ is a discrete time period of one day and represents a stage [S]. Then

$$\hat{n} \times RL \geq \sum_{N=1}^{[N]} RR[RT, N] \tag{17}$$

$$RL = 1, 2, \ldots, 5;\ (RL=1) < (RL=2) < , \ldots, (RL=5)$$

RL may represent any finite number of integer resource days, but only five alternative levels are allowed. The range of a job is the number of time periods (S) within which the job must be accomplished to avoid delaying the critical path. The range (R) is found by using a resource assignment of RL=5 and performing a critical path backward and forward pass to compute the early start and total float (TF) of each job N. The total float is used to determine the partitioned float (PF) of N. The range (R) is then equal to the duration (D) of N plus the partitioned float (PF)

$$R[N] = D[N] + PF[N] \tag{18}$$

and

$$D[O,N] = \frac{RR[N]}{RT[RL=5]},\ D[O] = \text{initial duration} \tag{19}$$

Because the maximum resource level is used for the first iteration, the initial duration is always a minimum and some jobs may be unnecessarily crashed. Concurrent requirements for a resource which resulted from network logic can create a circumstance where the concurrent jobs $\Sigma RR[RT,S]$ cannot be accomplished within the stage [S]. Then S must be extended to $\hat{n}$ which allows RL[RT] to accomplish $\Sigma N[RT]$.

$$\hat{n} = \frac{RR[RT,S]}{RL[RT]} \tag{20}$$

The work assigned to each stage S is $\frac{RR[S]}{RL[RT]}$, the work accomplished in each stage is $RL[RT]\hat{n}$. The assignment is the work required rather than the resources available and

$$\sum_{\hat{n}=1}^{\{\hat{n}\}} \gamma[\hat{n}] \geq \Sigma RR[RT,S] \tag{21}$$

Additionally, RR[RT, S] may be either a critical or non-critical plus critical requirement. The critical work required CRR[RT, S] must be completed or else the project duration will be extended. The NCRR[RT, S] may not extend the project duration if a time frame can be found where the ΣResource-days available is at least equal to the requirement after the critical requirement has been satisfied.

Objective Function

The objective function expresses the total cost of accomplishing the work required by all resources, and is stated by:

$$R = \sum_{RT=1}^{M} \sum_{S=1}^{Max} \left[COST[RT,N] + IDLE[RT,N] + CLC[RT,N] \right] + PCC[P] \qquad (22)$$

The sum of these costs for all resources is the construction resource cost for the project.

Summary

Conceptually, partitioned float is used to release non critical activities to expand their times of accomplishment to that of interrelated critical activities. Then, if any individual requirement is not accomplished within its own time frame the total project time is extended.

The adaptation of Dynamic Programming for the construction resource problem is seen to have a special connotation of "stage." Initially the stage is generated from the duration of an activity divided by the maximum number of resources of the proper type available each day. If subsequently, the stage can not be accomplished by any feasible number of resources, the stage is allowed to assume the alternative of a longer stage duration.

With these conditions established the total process is performed by the Binary Dynamic Program which is explained in Chapter V.

CHAPTER V

DYNAMIC MODEL APPLICATION

The construction resource management model has been formulated as a dynamic programming problem. The dynamic problem represents a non-linear, discrete programming situation which can be solved for multiple resource variables using the DDDP constrained state techniques.

The application of the model will be illustrated with two examples. First, a manual application will be shown which will serve to fully explain the necessary operations. Secondly, the example problem will be solved using data processing assistance.

Manual Model Application

The problem to be solved manually is shown by Figure 27. The problem consists of nine activities which share three resource types. Table 1 lists the activities along with the work required by each activity and the duration of the activity which results from application of the maximum resource type to its respective requirement. To this point, the data required is essentially equivalent to that necessary to perform a critical path analysis in its most simple form. The additional input data relates only to the resource types and consists of the levels deemed feasible by a scheduler, the cost of applying each level, the cost incurred as a result of changing from one level to another, the cost of having a

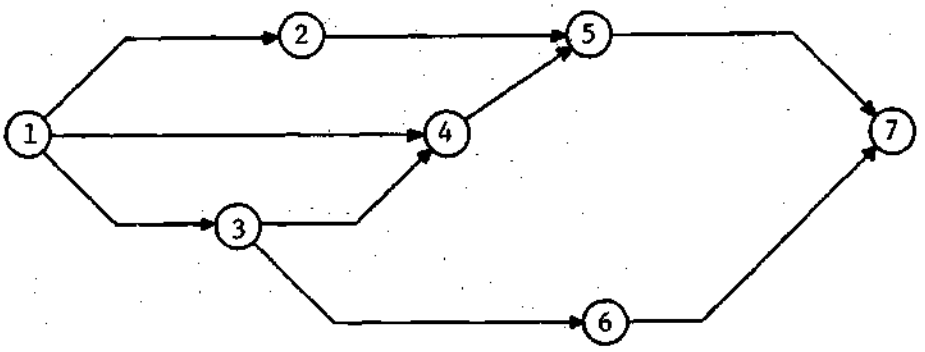


Figure 27. Example Problem Network.

Table 1. Example Problem Activity Data

Activity		Resource Type	Resource Days Required	Duration at Maximum Resource Level
I	J	RT(N)	RR(N)	DIJ(N)
1	2	B	8	1
1	3	B	7	1
1	4	A	14	2
2	5	C	6	1
3	4	A	8	2
3	6	B	15	2
4	5	C	6	1
5	7	B	8	1
6	7	A	6	1

Resource Type (RT)	Change Level Cost (CL)	Idle Cost (IDLE)
1	.5	1.0
2	1.0	1.5
3	.7	1.2

particular level available, but idle and the cost of project time. Table 2 lists this required data with the exception of project time which is selected to be $2.00 per day. The cost of changing resource levels and the idle cost is given as a unit type value in Table 1 and is applicable to both an increase or decrease in level.

Table 2. Resource Level Data for Example Problem Number One

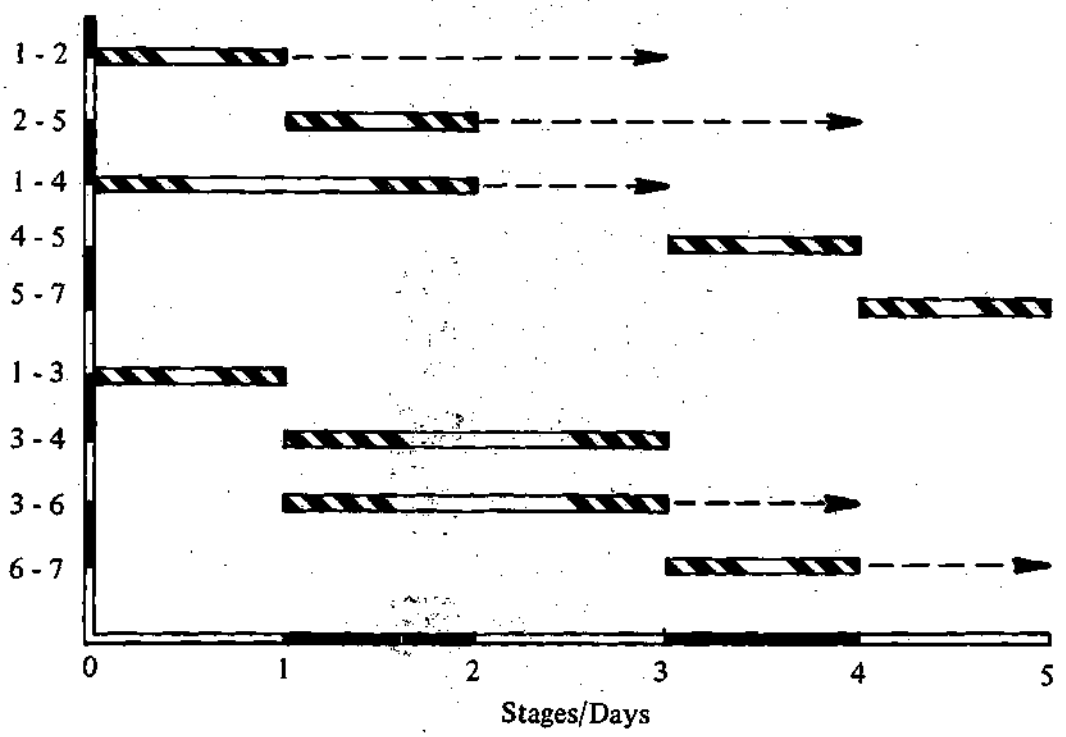
N	I	J	RT	RR	LEVELS					COST/LEVEL				
1	1	2	2	8	5	6	7	8	9	10	13	15	17	20
2	1	3	2	7	5	6	7	8	9	10	13	15	17	20
3	1	4	1	14	3	4	5	6	7	5	7	9	11	12
4	2	5	3	6	3	5	6	7	8	5	6	7	8	10
5	3	4	1	8	3	4	5	6	7	5	7	9	11	12
6	3	6	2	15	5	6	7	8	9	10	13	15	17	20
7	4	5	3	6	3	5	6	7	8	5	6	7	8	10
8	5	7	2	8	5	6	7	8	9	10	13	15	17	20
9	6	7	1	6	3	4	5	6	7	5	7	9	11	12

RT 1, 2, 3 = RT ABC respectively.

Segmenting Projects

The project is to be divided into discrete time intervals suitable for solution by dynamic programming. To accomplish this, the early start and late start times which result from a conventional critical path analysis are needed to map the network logic to a specific time frame.

Figure 28 is a time scaled representation of the logic network of Figure 27 and illustrates the origin of the daily resource requirements shown by the table accompanying the figure and the CP data shown in Table 3.



RESOURCE TYPE REQUIREMENTS/STAGE

RESOURCE TYPE [RT]		STAGE				
		1	2	3	4	5
1	CRITICAL	0	4	4	0	0
	TOTAL	7	11	4	6	0
2	CRITICAL	7	0	0	0	8
	TOTAL	15	7.5	7.5	0	8
3	CRITICAL	0	0	0	6	0
	TOTAL	0	6	0	6	0

Figure 28. Time Scaled Resource Requirements for Example Problem.

The original critical path calculations were performed with the durations set to that duration which results from maximum application of the type resource level available. The transition from precedence logic to resource logic requires the consideration of accomplishing a given amount of work in a specific time span set by the precedence logic. Consequently, when a total resource type requirement must be accomplished between t_1 and t_2 the volume of the resource application diagram is constrained to one of the feasible resource levels, the range from t_1 to t_2 must be adjustable to obtain the resource required. Figure 29 illustrates this concept.

Table 3. Table of Critical Path Results for Example Problem

N	I	J	ES	LS	EF	LF	FF	TF
1	1	2	0	2	1	3	0	2
2	1	3	0	0	1	1	0	0
3	1	4	0	1	2	3	1	1
4	2	5	1	3	2	4	2	2
5	3	4	1	1	3	3	0	0
6	3	6	1	2	3	4	0	1
7	4	5	3	3	4	4	0	0
8	5	7	4	4	5	5	0	0
9	6	7	3	4	4	5	1	1

The original critical path schedule is used to determine the amount of resource required between times t_1 and t_2 and a choice of using either early start (ES), late start (LS) or some arbitrary start must be made. Due to the obvious and noted disadvantage of all choices, the noncritical activities are expanded by an

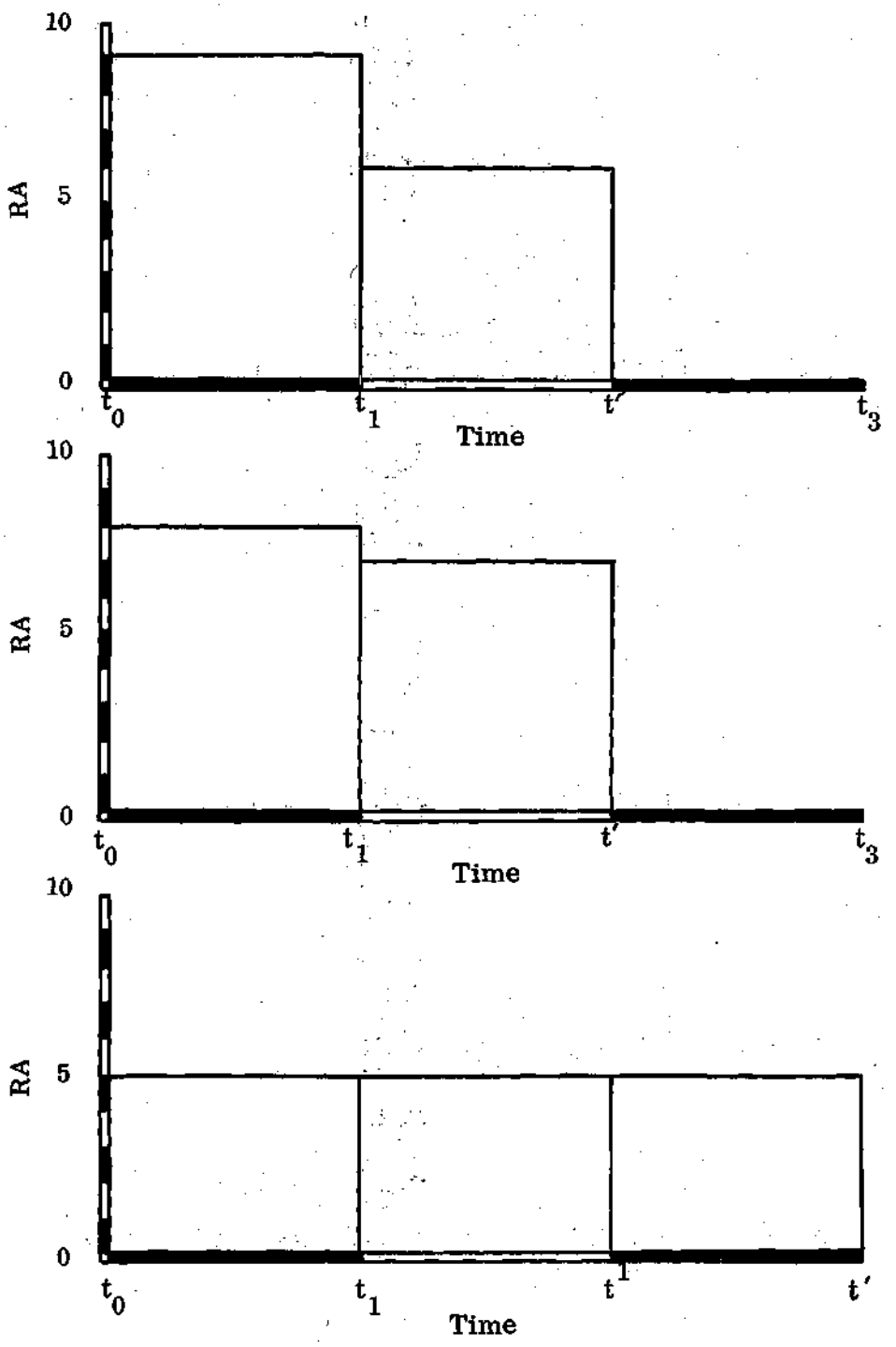


Figure 29. Resource Application Diagram.

amount equal to their partitioned float. The partitioned float is shown by Table 4 and although initially nonexistent in this example problem, the partitioned float will become a factor in a later step.

Table 4. Partitioned Float for Example Problem

I	J	RR[IJ]	RR[CHAIN]	TF	PF
1	2	8.00	22.00	2	0
1	3	7.00	29.00	0	0
1	4	14.00	28.00	1	0
2	5	6.00	22.00	2	0
3	4	8.00	29.00	0	0
3	6	15.00	28.00	1	0
4	5	6.00	29.00	0	0
5	7	8.00	29.00	0	0
6	7	6.00	28.00	1	0

The partitioned float is used to modify the durations of the non-critical activities and the critical path calculations are revised. In this example, the results of the critical path calculations remain unchanged from those given by Table 3.

The project can now be segmented into one day increments and the segment transformed to "stage" logic. The selection of one day stages, where a stage is designated ns, results from the previous discussion of the practical ability to change resource decisions daily, combined with the impractical nature of scheduling time intervals shorter than one day. Additionally, use of the

minimum activity schedule length for the stage duration simplifies the search for interdependent resource logic as will be seen in later steps of the solution process.

Determining Stage Duration

With the stage resource requirement established, the stage duration becomes a function of the resource decision and the decision to schedule or not schedule the non-critical resource requirements. Although the stages were originally segmented into one day durations, the resource requirement was based upon application of the maximum respective resource level. Subsequently, it is possible that the logic of the network will combine with the limited resource availability to create a condition where the stage can no longer be accomplished in one day. Further, the performance of all the required work within the original stage duration may not be the optimum decision. Consequently, the specified feasible resource levels must be applied in a series that provides at least as much effort as is required by the stage. Whether the resulting series is level or variable should and does depend upon the cost of the decision. The alternate assignments which are available consist of the input resource levels in serial sequences sufficient to satisfy the requirement for the resource type. The data given in Table 2 shows that resource type two has five levels which are 5, 6, 7, 8, and 9 units of the resource type. The serial sequence must be constructed from these alternatives. Stage one of the example problem requires 15 units of resource type two. By observation, the following series, although not necessarily optimal, are available to satisfy this requirement:

5 - 5 - 5

6 - 6 - 6

7 - 8

8 - 8

9 - 9

None of the series can accomplish the stage in one day since 9 is the maximum resource level available. The desired series also has the requirement of exhibiting a minimum cost. Since the result of obtaining this initial stage duration will be input to further processing, an absolute optimum is not required and any effort expended in reaching a better initial solution should be less than that of the subsequent, iterative search procedure. Consequently, the initial stage duration (ST) is derived by testing each level in series for two attributes: (1) greater than the resource type required and (2), a minimum cost. The sum of the levels allocated provides the comparison with the resource required and the minimum cost is found by summing the cost of each level (COST), plus the cost of changing levels (CLC), plus a penalty value for using more than one stage (DAILY) and the value of leaving any resource available in an idle state (IDLE). Using the input data for resource type two and the project daily cost of $2.00 the following series illustrate these costs:

Series	Cost		CLC		Idle		Daily		Total
5,5,5	30	+	0	+	18	+	4	=	$52.00
7,8	32	+	1.0	+	4.5	+	2	=	$39.50
9,6	33	+	4.5	+	3	+	2	=	$42.50

With these costs as candidates, the allocation of 7 units on one day followed by

8 units on a second day shows promise of being an optimum assignment. Selecting an optimum for one stage does not produce an optimum for the project due to the uncalculated requirement in future stages all of which are interdependent. The dependency between stages consists of the change in level of the resource allocated. For example, if one stage n could be satisfied with an allocation of 5 and the stage n+1 satisfied with 7, the stages are dependent to the extent of changing from the allocation of 5 units to the allocation of 7 units of the resource type. In order to approach a minimum cost for the project, a minimum cost series ending with each of the levels and a minimum cost series beginning with each level are required. With 5 levels for each resource type Table 5 gives the set of allocations and costs which provide initial candidates for optimums in stage one for resource type two.

These candidates represent an optimum for the total resource requirement (RR). In many cases, the application of the partitioned float causes each activity to become so near critical that a practical distinction is beyond the degree of accuracy of the basic data. However, to maintain the capability of performing the non-critical requirement without extending the critical path an additional set of resource allocation series (RA) sufficient for only the critical resource requirement is generated and illustrated by Table 6. Similar series are prepared for each resource type in each stage.

Trial Solution

The series of allocation levels available for each resource type are now

Table 5. Set of Optimum Candidates Stage One,

Resource Type Two

NS	RR	RESOURCE ALLOCATED			COST	ST
1	15	6	9		42.50	2
1	15	9	6		42.50	2
1	15	7	8		39.50	2
1	15	8	7		39.50	2
1	15	5	5	5	52.00	3

Table 6. Set of Optimum Candidates for the Critical Requirements

of Resource Type Two, Stage One

NS	RR	RESOURCE ALLOCATED		COST	ST
1	7	7		18.00	1
1	7	8		18.50	1
1	7	9		20.00	1
1	7	5	5	34.00	2
1	7	6	5	36.50	2

reordered according to their costs and three values are selected from both the critical and total series. For the first cycle, it is desirable to test a wide corridor and gradually narrow the corridor towards the optimum in each cycle. The first trial solution consists of six values for each resource type in each stage. For the example problem, the costs of resource type two, stage one and their associated resource allocations follow in Table 7.

Table 7. Initial Values for Stage One, Resource Type Two

	CRITICAL			TOTAL		
COSTS	18.00	20.00	36.50	29.50	42.50	52.00
RA	7	9	5,6	8,7	9,6	5,5,5
ST	1	1	2	2	2	3

Similar values are extracted for all resource types and the solution process proceeds to the binary dynamic program.

Recursive Analysis

Regardless of the optimum solution in a current stage, the alternative of making a different decision must be maintained until the possibilities of all stages have been scanned. Normally, the DP procedure provides an excellent fit to these requirements. However, the necessity to perform a "dual" process on both the critical and non critical requirements demands modification of normal tabular procedures. In deference to the higher order exponential terminology (D^3P)

already established by Heidari, Chow and Meredith, the algorithm will be simply referred to as a binary variable procedure for tabular DP.

The binary dynamic program (BDP) will operate on both the critical and non critical resource requirements simultaneously. The solution of the first stage of the BDP is trivial since it consists of the data itself. The BDP solution procedure for stages greater than one is shown by Table 8.

Table 8. BDP Solution Procedure, Stages Greater than One.

RA		$RA_{n-1,1}$	$RA_{n-1,2}$	...	$RA_{n-1,6}$		
	f^*	$f^*_{n-1,1}$	$f^*_{n-1,2}$	...	$f^*_{n-1,6}$	$F^*_{n,J,I}$	$RA_{n,J}$
$RA_{n,1}$	$f^*_{n,1}$	$F_{n,1,1}$	$F_{n,1,2}$	...	$F_{n,1,6}$	$\min_{I=1,6} F_{n,1,I}$	$RA_{n,1}$
$RA_{n,2}$	$f^*_{n,2}$	.	.		.		
.	.	.	.		.		
.	.	.	.		.		
.	.	.	.		.		
$RA_{n,6}$	$f^*_{n,6}$	$F_{n,6,1}$	$F_{n,6,2}$	...	$F_{n,6,6}$	$\min F_{n,6}$	$RA_{n,6}$

RA[n] = RESOURCE ALLOCATIONS CURRENT STAGE

f^*[n] = COST OF RESOURCE ALLOCATION, CURRENT STAGE

F[n,1,1] = f^*[n-1,1] + f^*[n,1] + C

C = CHANGE LEVEL COST FROM RA[n-1] to RA[n]

I = INDEX OF THE COLUMN GENERATING THE MIN. F[n,1,I]

Noting that the table consists of six values from each stage and that the first three are critical and the following three are total values, the following relationship is seen to exist as shown in Table 9.

Table 9. Quadrant Relationship of Critical and Total Resource Allocation

	1	2	3	4	5	6
1		I			II	
2		Critical Cost $_{n-1}$			Total Cost $_{n-1}$	
3		+ Critical Cost $_{n}$			+ Critical $_{n}$	
4		IV			III	
5		Critical Cost $_{n-1}$			Total Cost $_{n-1}$	
6		+ Total Cost $_{n}$			+ Total Cost $_{n}$	

To provide for all alternatives of critical and total combinations, the BDP cannot select the minimum of total costs as compared to critical costs. Doing so would always result in selection of the critical and neglect the alternative of accomplishing the non critical in a previous stage. Consequently, the minimum of rows 1 through 3 and columns 1 through 3 are tested for a minimum cost. Rows 1 through 3 are saved with the minimum selected and the rows 4, 5 and 6 are first tested to greater than or equal to the resource required and the remaining candidates are compared row by row for a minimum between quadrants II, III, and IV.

This is shown by the following:

$$F^*_{n,1} = \min \{(F_{n,1,1}), \ldots, (F_{n,1,3})\} \tag{23}$$

$$F^*_{n,2} = \min \{(F_{n,2,1}), \ldots, (F_{n,2,3})\} \tag{24}$$

$$F^*_{n,3} = \min \{(F_{n,3,1}), \ldots, (F_{n,3,3})\} \tag{25}$$

$$F^*_{n,4} = \min [(F_{n,1,4}) \ldots (F_{n,1,6}), (F_{n,4,1}), \ldots, (F_{n,4,6})] \tag{26}$$

$$F^*_{n,5} = \min [(F_{n,2,4}), \ldots, (F_{n,2,6}), (F_{n,5,1}), \ldots, (F_{n,5,6})] \tag{27}$$

$$F^*_{n,6} = \min [(F_{n,3,4}), \ldots, (F_{n,3,6}), (F_{n,6,1}), \ldots, (F_{n,6,6})] \tag{28}$$

Subject to: $RA_{n,I,J} + \sum_{n=1}^{N} RA_{n-1}, \text{ INDEX} \geq \sum_{n=1}^{N} \text{TOTAL RR},$

$I = 1, 6$

$J = 1, 6$

Applying these formulae to the example problem yields Table 10 for resource type two, stage two.

Table 10. Minimum Costs for Resource Type Two, Stage Two

$F^*[2,J]$	RA[2,J]	RA[1,J]
25.0	00.0000.0000.0000.0000.000	7.0000.0000.0000.0000.000
25.0	00.0000.0000.0000.0000.000	7.0000.0000.0000.0000.000
25.0	00.0000.0000.0000.0000.000	7.0000.0000.0000.0000.000
59.0	08.0000.0000.0000.0000.000	8.0007.0000.0000.0000.000
75.5	05.0005.0000.0000.0000.000	8.0007.0000.0000.0000.000
75.5	00.0000.0000.0005.0007.000	0.0000.0000.0005.0006.000

This data resulted from the minimum of the interior cost array shown by Table 11.

Table 11. Interior Cost Array Resource Type Two, Stage Two

F[2,J,I]

*25.00000	29.00000	42.50000	46.50000	48.50000	57.00000
*25.00000	29.00000	42.50000	46.50000	48.50000	57.00000
*25.00000	29.00000	42.50000	46.50000	48.50000	57.00000
37.50000	39.50000	57.00000	*59.00000	63.00000	73.50000
54.00000	58.00000	71.50000	*75.50000	77.50000	86.00000
58.00000	62.00000	*75.50000	79.50000	81.50000	90.00000

* Indicates minimum selected.

The binary dynamic procedure is performed for all resource types in each stage and the process is repeated for each stage to the last.

Traceback for Optimum Allocations

When all resource types have been processed through the BDP of the

last stage, it is necessary to determine the allocations which resulted in the minimum cost for the last stage (OF*). This value in the last stage is selected directly as the minimum cost available.

$$OF^*_{NS} = \min(F^*_{NS,J}) \quad J=1,6 \tag{29}$$

Associated with this minimum is an index I which points to the J of the $F^*_{NS-1,J}$ which, when added to the $F_{NS,J}$ of the current stage resulted in the OF^*_{NS}. Figure 30 shows this procedure out of context with the example problem.

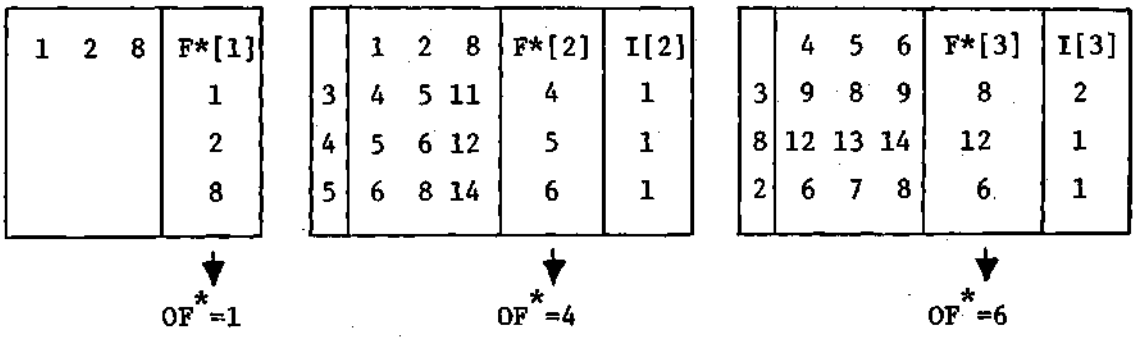


Figure 30. Simplified Traceback Procedure

DDDP2

The secondary discrete differential procedure uses the OF* of each stage and each resource type as a focal point to converge the subsequent three values from the feasible resource series. For example, if resource type two, stage one of the example problem required five series of feasible allocations as follows:

1	8,7
2	7,8
3	6,9
4	9,6
5	5,5,5

If the initial trial solution was series 1, 3 and 5 and OF^*_{ns} resulted from series 3, then series 1 and 5 would be moved towards 3. If OF^*_{ns} resulted from value 5, then 1 and 3 would be moved towards value 5. After new values are chosen, a new cycle begins and the binary dynamic procedure is performed again using the new values. The system is allowed to cycle until convergence is attained or until ten cycles have been completed.

The expansion of stages by cycling different values through the BDP perturbs the system. In particular, some resource types may now require stage durations greater than the original one day. Should this be the case, the original critical path is no longer valid. To allow the system to regain equilibrium, the activity durations in each stage must be lengthened by an amount equal to the least cost stage duration (ST) minus the original stage duration of one day. The new durations are input to the critical path procedure and a new iteration is started through the same steps as were performed for the initial values. The critical path resulting from updating the stage duration of the example problem, iteration two, is shown by Table 12. It is noted that the total float has now been increased to the point that partitioned float can be applied to some activities. Table 13 shows the partitioned float and the critical path which evolved from

Table 12. Critical Path Data with Revised Activity Durations

N	I	J	ES	LS	EF	LF	FF	TF
1	1	2	0	3	2	5	0	3
2	1	3	0	0	2	2	0	0
3	1	4	0	2	3	5	2	2
4	2	5	2	5	3	6	3	3
5	3	4	2	2	5	5	0	0
6	3	6	2	4	4	6	0	2
7	4	5	5	5	6	6	0	0
8	5	7	6	6	7	7	0	0
9	6	7	4	6	5	7	2	2

extending the durations of those activities having partitioned float.

Iterations of the procedure are continued until the restrictive convergence criteria is reached or until ten iterations have been performed. The convergence of an iteration is based upon the sum of minimum F^*_{ns} of all resource types of the last stage, last cycle (CF*)

$$CF^*_{IT} = \sum_{RT=1}^{max} OF^*_{ns,RT}, \quad ns=max \tag{30}$$

Figure 31 shows this procedure in block diagram format.

Computer Solution

The machine assisted solution of the multiple resource problem takes advantage of the rapid calculation capability to reduce the dimensionality of the

Table 13. Tables of Partitioned Float and Critical Path

I	J	RR[IJ]	RR[CHAIN]	TF	PF
1	2	8.000	22.00	3	1
1	3	7.000	29.00	0	0
1	4	14.000	28.00	2	1
2	5	6.000	22.00	3	0
3	4	8.000	29.00	0	0
3	6	15.000	28.00	2	1
4	5	6.000	29.00	0	0
5	7	8.000	29.00	0	0
6	7	6.000	28.00	2	0

I	J	ES	LS	EF	LF	FF	TF
1	2	0	2	3	5	0	2
1	3	0	0	2	2	0	0
1	4	0	1	4	5	1	1
2	5	3	5	4	6	2	2
3	4	2	2	5	5	0	0
3	6	2	3	5	6	0	1
4	5	5	5	6	6	0	0
5	7	6	6	7	7	0	0
6	7	5	6	6	7	1	1

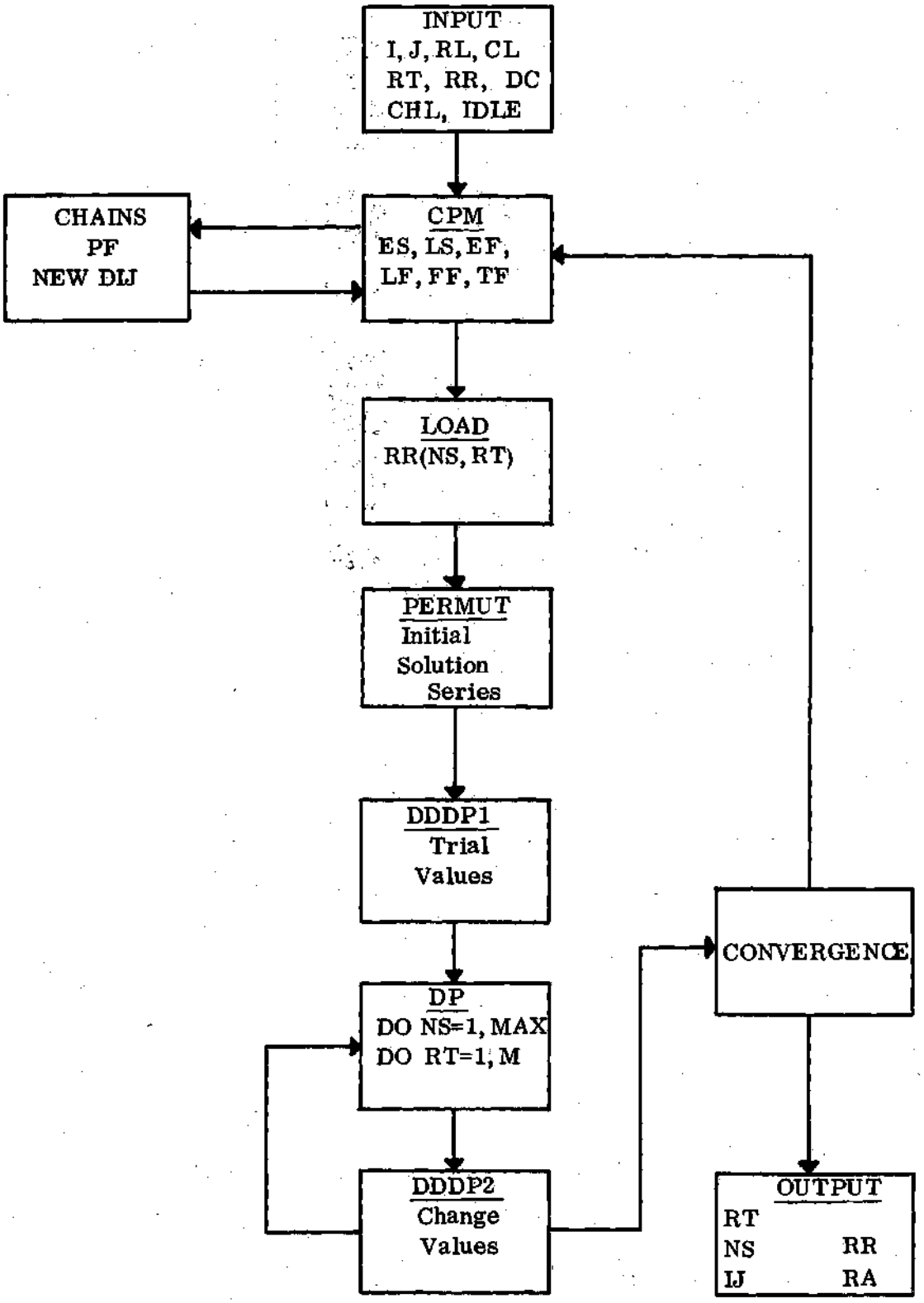


Figure 31. RSP Computer Solution Sequence.

problem by working only on the current values within a corridor and iterating the solution to obtain convergence from all values. Since the initial solution represents a minimum infeasible time solution, the subsequent resource constrained solutions serve to extend the critical path by applying the resources in available quantities. Essentially, each partial path is extended to a constrained resource, feasible length and all remaining shorter parallel paths are brought to isochronal incidence at a minimal cost. The computer solution works on the serial and multiple resource problem simultaneously. This procedure provides the alternative of ischronous stages for all resources while insuring, through constraints, that the interconnecting resource chains will maintain their proper precedence.

Once formulated, the manual computation although exhausting, is straight forward. Alternately, the computer solution is seemingly quite complicated as shown by the block diagram depicting the step procedure required for the computer code, Figure 31.

Input Data

The input requirements are oriented to the activity and the resource type. For the activity, the I-J activity identifier, the single type of resource for the activity and the resource days required to complete the work described by the activity must be input. To describe the type of resource, the feasible levels must be identified along with the respective cost of supplying these levels. For each type of resource, the cost of changing the level per resource unit and the idle cost per unit must also be determined. The code is dimensioned for a maximum of ten resource types of five levels each and the schedule length is designed

not to exceed 66 days or three working months. The additional data required is the daily cost of the project which may be either the delay damage cost, the project indirect costs, or a combination of both categories of cost. Attention must be paid to the application of delay damage costs upon a schedule which is less than the date on which delay damages will begin to accrue. The program does not automatically assess the damages after a specified date. Sample input data for the example problem is shown by Table 14.

Table 14. Sample Input Data For the Example Problem

```
1.   090305000000002.0
2.   00100200200800500600700800901001301501702O
3.   00100300200700500600700800901001301501702O
4.   00100400101400300400500600700500700901101 2
5.   00200500300600300500600700800500600700001O
6.   00300400100800300400500600700500700901101 2
7.   00300600201500500600700800901001301501702O
8.   00400500300600300500600700800500600700801O
9.   00500700200800500600700800901001301501702O
10.  00600700100600300400500600700500700901101 2
11.  0000000.500000001.0
12.  0000001.000000001.5
13.  0000000.700000001.2
```

Critical Path Sub-Routine

The critical path sub-routine uses the input data to construct a critical path with maximum resource levels. Those activities which are then non critical are extended to the end of their range flag or as far as possible by adding the respective partitioned float to the duration. The resulting activity duration is combined with the early start time (ES) from a regenerated critical path to form the basis for the stage by stage resource requirements for each resource type. The cost of the project due to resource allocation includes the total cost of all resource allocations plus the daily project cost for the project duration. The basic critical path code Appendix A is adapted from the Association of Computing Machinery, Algorithm No. 40 and could consist of any code which provides the standard CPM data.

Chains Sub-routine

The sub-routine assigns all activities to a respective chain. The total float from the CP sub-routine and the resource-days per activity are used to calculate the partitioned float plus duration which is then set equal to the Range (R) or new duration of each activity. Appendix B shows the code necessary for identification of the chains of a network.

Load Sub-routine

The Load program uses the early start (ES) and duration (DLJ) calculated by the Critical Path routine to load the stages with the resource required (RRQD) for each stage. The required information is kept in three arrays: LOD (NS, RT, 1) gives the critical RR for each stage by resource type (RT), LOD (NS, RT, 2)

gives the total RR for each resource type and LOD(NS, RT, 3) contains the IJ of all activities ending in a stage (NS). Appendix C shows this code.

PERMUT Sub-Routine

The accomplishment of the work required by each stage can only be done by using a series of level or varying resource assignments. Initially, the maximum resource assignment is adequate for a single activity, however, if resource requirements become simultaneous it may be impossible to accomplish a given stage in one day. The PERMUT sub-routine calculates a minimum series of feasible assignments of the input levels to perform each stage. The transition between stages is cost dependent upon leaving and entering resource levels, consequently an alternative series of leaving each stage at all levels and entering each stage at all levels is provided. In addition, the series are separated according to critical and non critical requirements for each stage, since the non critical requirement may be slipped to the end of its range. The program selects three values from the ordered series of data; the lowest cost, the mean cost, and the highest cost. These values are passed to the BDP routine. This function is performed by the code listed as Appendix D.

BDP Sub-Routine

The BDP routine uses six values from DDP1. The first three values are associated with the critical requirement; the second three are associated with the total resource requirement for each stage and the resource type. The cost of the six values are combined between stages by creating a seven by seven array, A (NS, RT, J, I). The costs for a current stage result from DDP1 while those of

the previous stage are the row-quadrant values selected as minimums. Each cost is tested to determine that the associated resource allocation when added to previous allocations is at least equal to the resource required for the current and all previous resource type requirements. Quadrant I is first searched for a minimum greater than or equal to the resource required. If a value meeting both requirements does not exist, the row minimum FSTR is set equal to FSTR + 10^5. The minimums from quadrants II, III, and IV are only selected from costs having adequate resource allocations. It is noted that the "A" array is not modified by adding 10^5 thereby retaining the capability to become feasible in a later stage. Appendix E lists the code for this sub-routine.

DDDP2 Sub-Routine

DDDP2 performs the function of selecting trial solutions for cycles greater than one. The trial values are selected from the feasible series calculated by the Permut sub-routine. The most optimal solution is saved and the corridor is narrowed toward the optimal as previously explained. The procedure always maintains one alternate value on either side of the optimum unless the optimum is a boundary. The sub-routine is shown by Appendix F.

DURMOD Sub-Routine

The DURMOD routine, Appendix G, is used to modify the activity durations to the extent of the optimum stage time required to perform the resource requirements of each stage. The optimum stage times for each stage and resource type are found by tracing the minimum cost backwards from the last stage to identify the value contributing to the minimum cost in each stage. The stage time

associated with this optimum cost is the optimum stage time for the resource type. The activities existing in each stage are extended by the optimum stage time minus one and a new critical path calculation is performed. Durmod is called after all cycles are complete in each iteration.

RESOURCE SYNTHESIS PROGRAM - MAIN

The main program controls the sub-routines, stores desired data, and generates the output in the required format. The code for this control is shown by Appendix H.

CHAPTER VI

EVALUATION OF MODEL

The algorithm was evaluated with the fortran code listed in Appendices A through H. The objective of the evaluation was primarily to develop the range of model applications within which the algorithm can perform. The basic network used for evaluation was the same as that of the example problem explained in Chapter VI. This network was selected for its simplicity as well as a definitive ability for repression of results. The model was evaluated for the following attributes:

1. Minimum cost resource allocations with limited resources
2. Minimum time schedules with limited resources
3. Maximum time schedules with minimum resource levels
4. Level resource allocation schedules

Minimum Cost Resource Allocation

The algorithm evaluates the cost of allocating feasible levels plus the costs of changing levels, idle resources and the cost of project time. The series of allocations, each at least as great as the critical or total resource requirement for a stage and resource type, make up a set of alternatives for the dynamic program. The dynamic program eliminates those alternatives which must subsequently prove more costly than the remaining candidates for optimums. The

procedure is based upon the premise that a specified amount of work requiring expenditure of one resource type may be accomplished in one or more days by applying one or more resource levels in a sequence that results in a minimum total cost.

The algorithm was tested with the data given in Table 14. The critical path sub-routine performed perfectly in all tests on this and other data. The addition of the partitioned float failed to have a significant effect upon the durations of the sample problem activities. This result is attributed to the limitation of activity durations to increments of the least whole day. Working with a sample problem of short total duration an activity, although having total float, did not have a large enough share of that float to obtain a share of at least one day of partitioned float. In practice this effect of unused float will exist, but will be of much less consequence since the capability of estimating durations with an accuracy of less than one day is unrealistic.

The initial values from the Permut subroutine proved more accurate than anticipated. Examination of the data shown by Figure 33 revealed that iteration convergence was attained on the second iteration. The reason for convergence being obviously not so much due to the efficiency of the Permut algorithm, but rather due to the manner in which the procedure approached the optimum. Initially the durations of the activities were set to a feasible length by dividing the resource requirement by the maximum level available. When the logic of the network created parallel resource type requirements Permut again adjusted to the duration of each activity by determining a stage time which would allow accomplishment of

the requirement by at least the maximum resource level. Figure 32 illustrates this procedure.

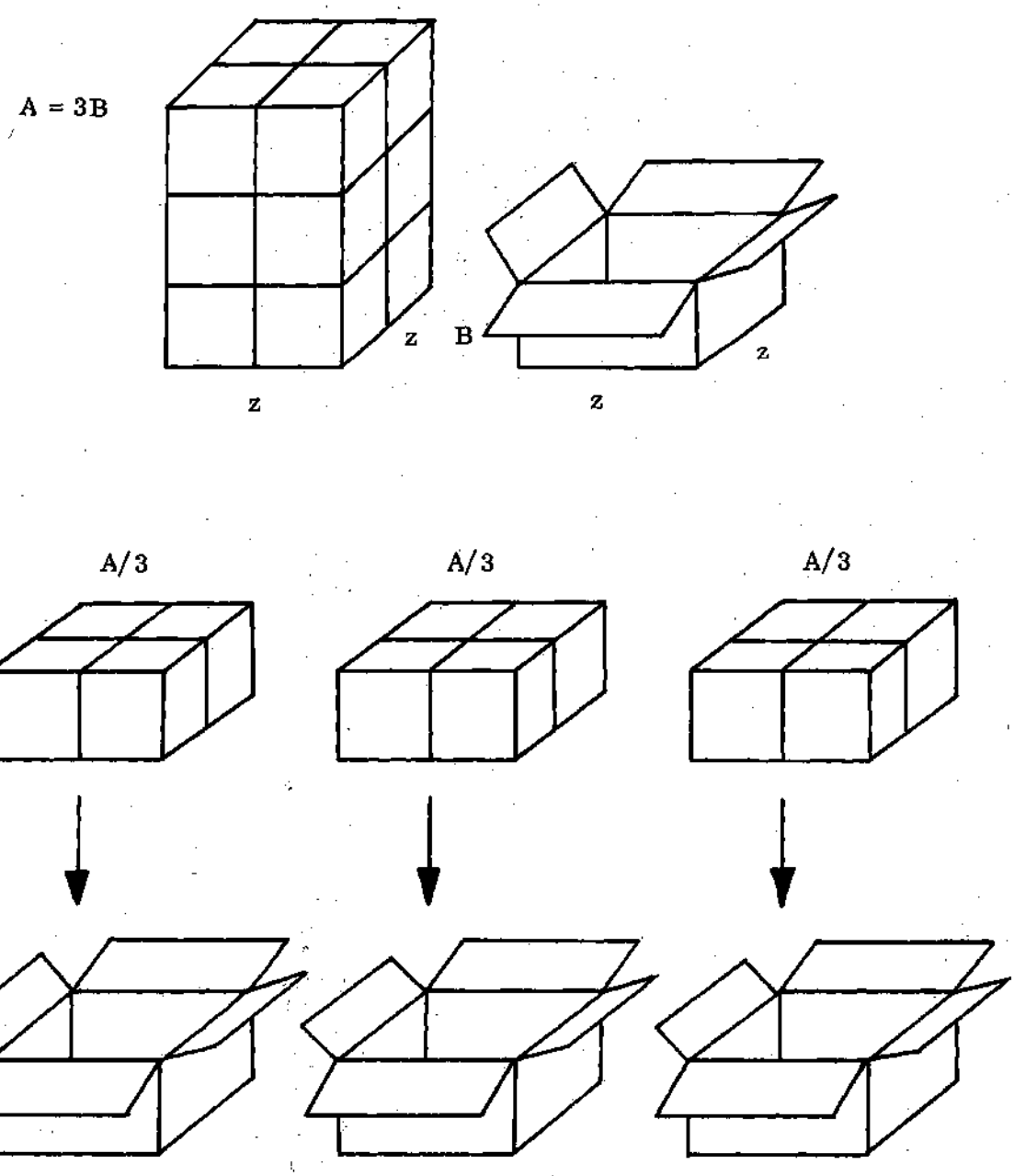


Figure 32. Illustration of Permut Process.

On the first iteration, A is a stack of material too large to be placed in box B. Some procedure reduces the size of this material to less than that of the box.

On iteration two, the size of box can be adjusted to save costs as long as it holds the material. Such a procedure is similar to the operation of Permut and is quite likely to find a solution sufficiently close to the values from which it was generated.

Figures 33 through 38 show the results of the allocations for each of three resource types for progressive cycles and iterations. Resource type one converged to its minimum cost on cycle 4, iteration 2. Figure 34 shows a total allocation of 28 resource days which is not excessive in comparison to the resource day requirement. However, an extra allocation may occur when the cost of having a zero allocation is greater than the cost of allowing the resource allocation to remain level. This procedure departs from optimality since this part of the code is a fix up routine that checks for a zero allocation and then checks for either a level cost or a zero cost depending upon whichever is least costly.

Resource type two converged on cycle 4, iteration 2. This resource type required the longest schedule which implies that the work required for the resource type diverged from the resource availability to a greater degree than did the other types. While this resource is the dominant type, no degree of resource criticality is implied since the type one resource may be inordinately expensive when compared to resource type two.

Resource type three, having a limited intermittent requirement was of

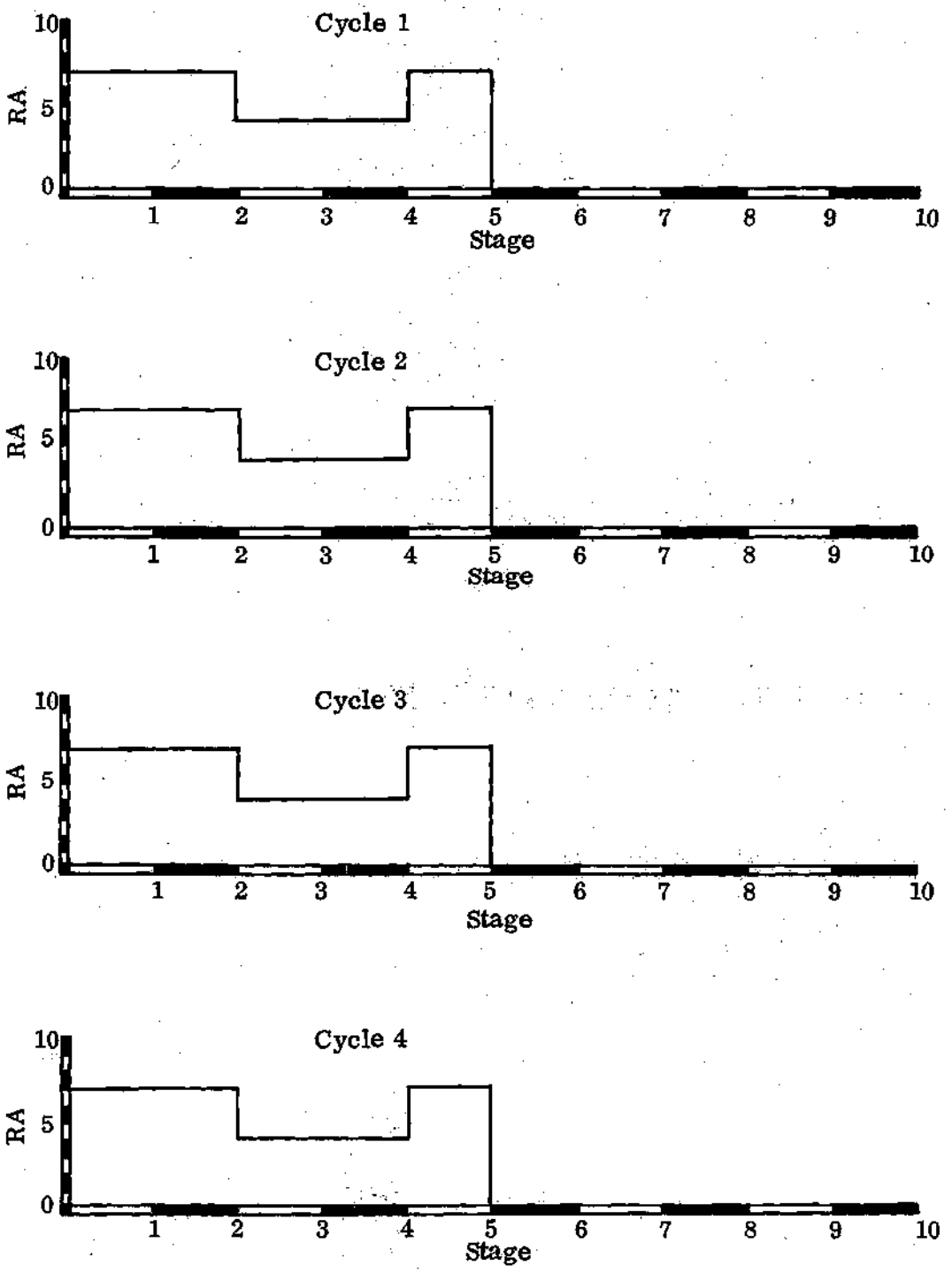


Figure 33. Resource Allocation for Resource Type One, Iteration One, Cycle One through Four.

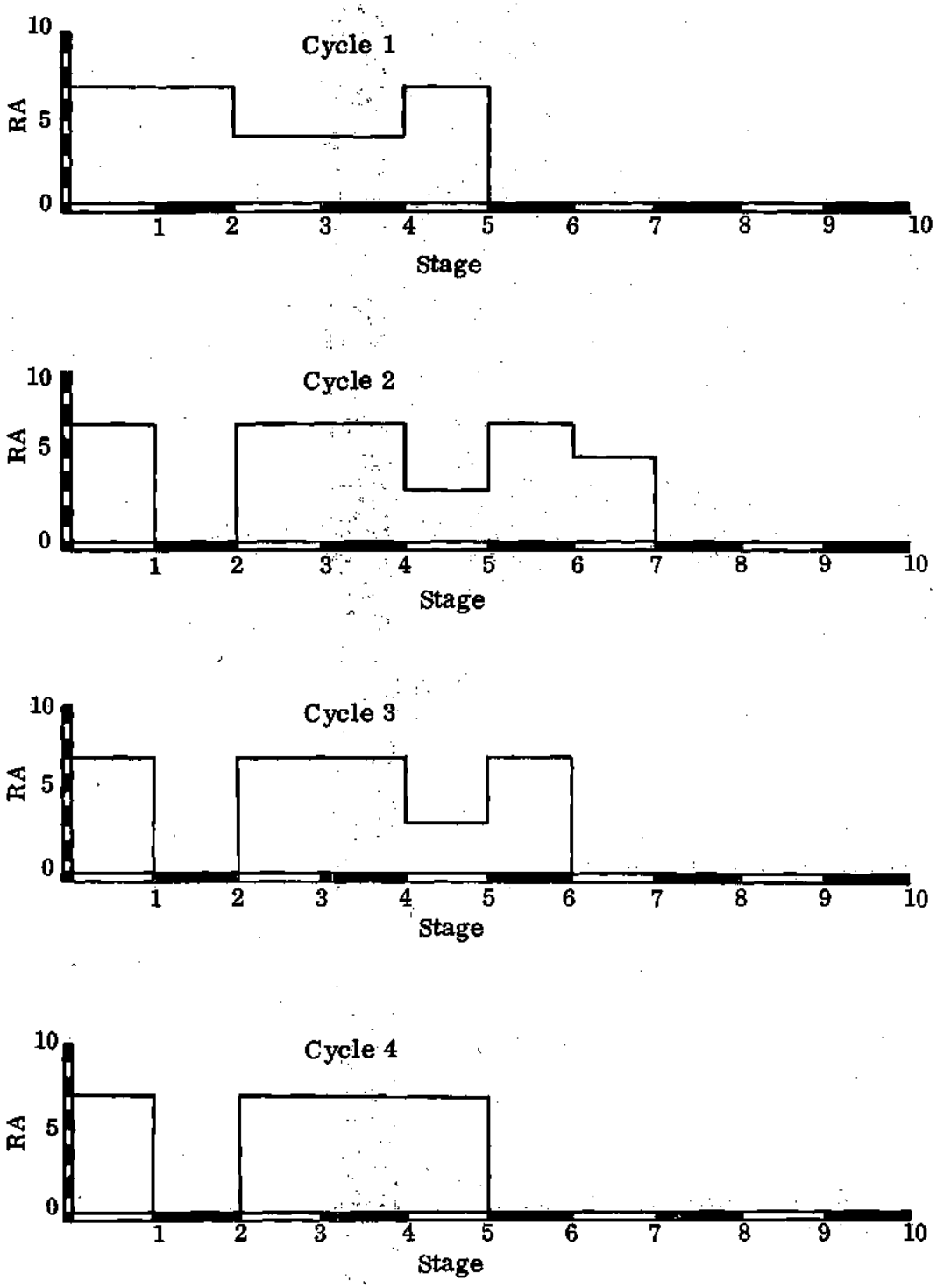


Figure 34. Resource Allocation for Resource Type One, Iteration Two, Cycle One through Four.

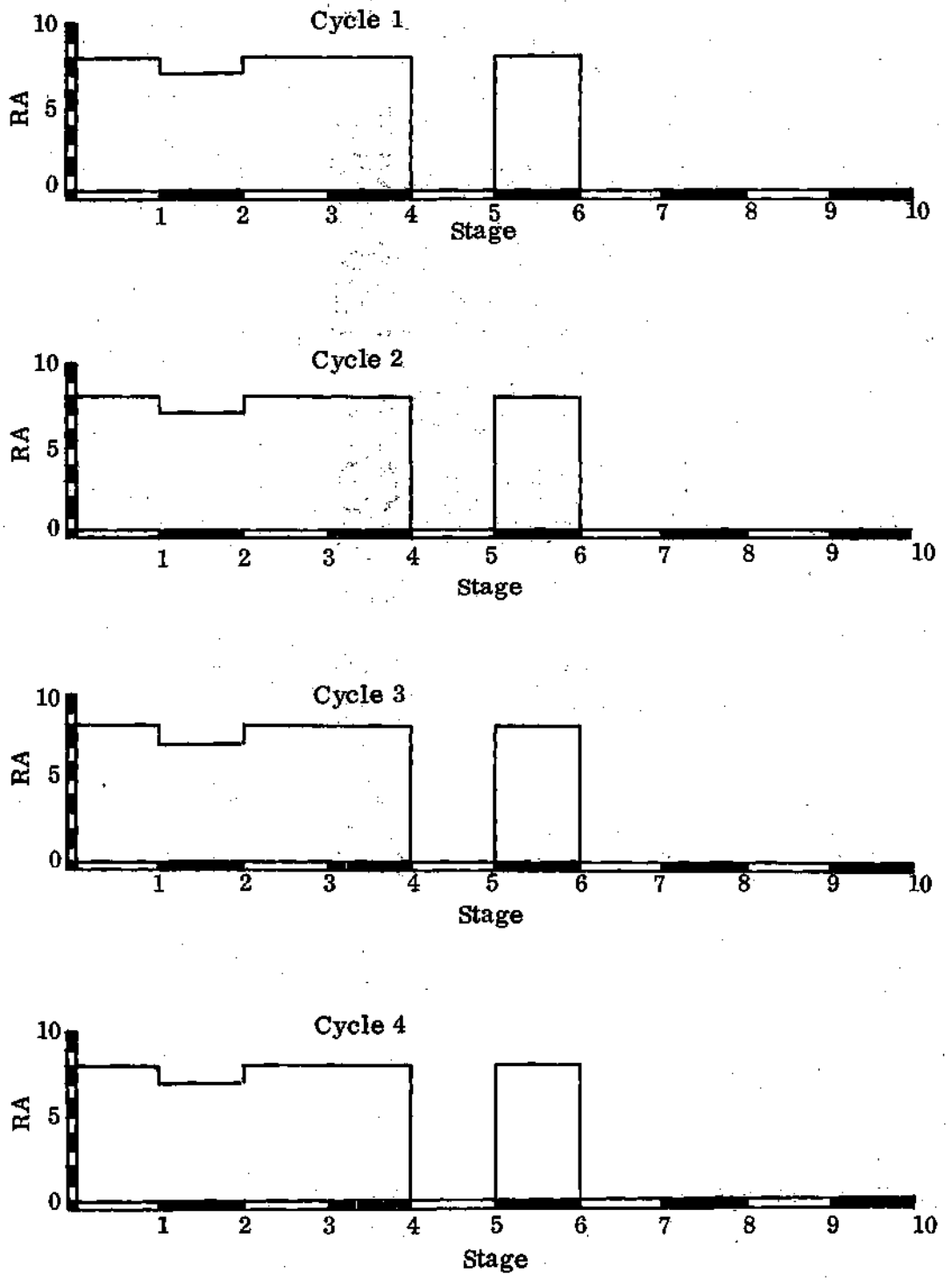


Figure 35. Resource Allocation for Resource Type Two, Iteration One, Cycle One through Four.

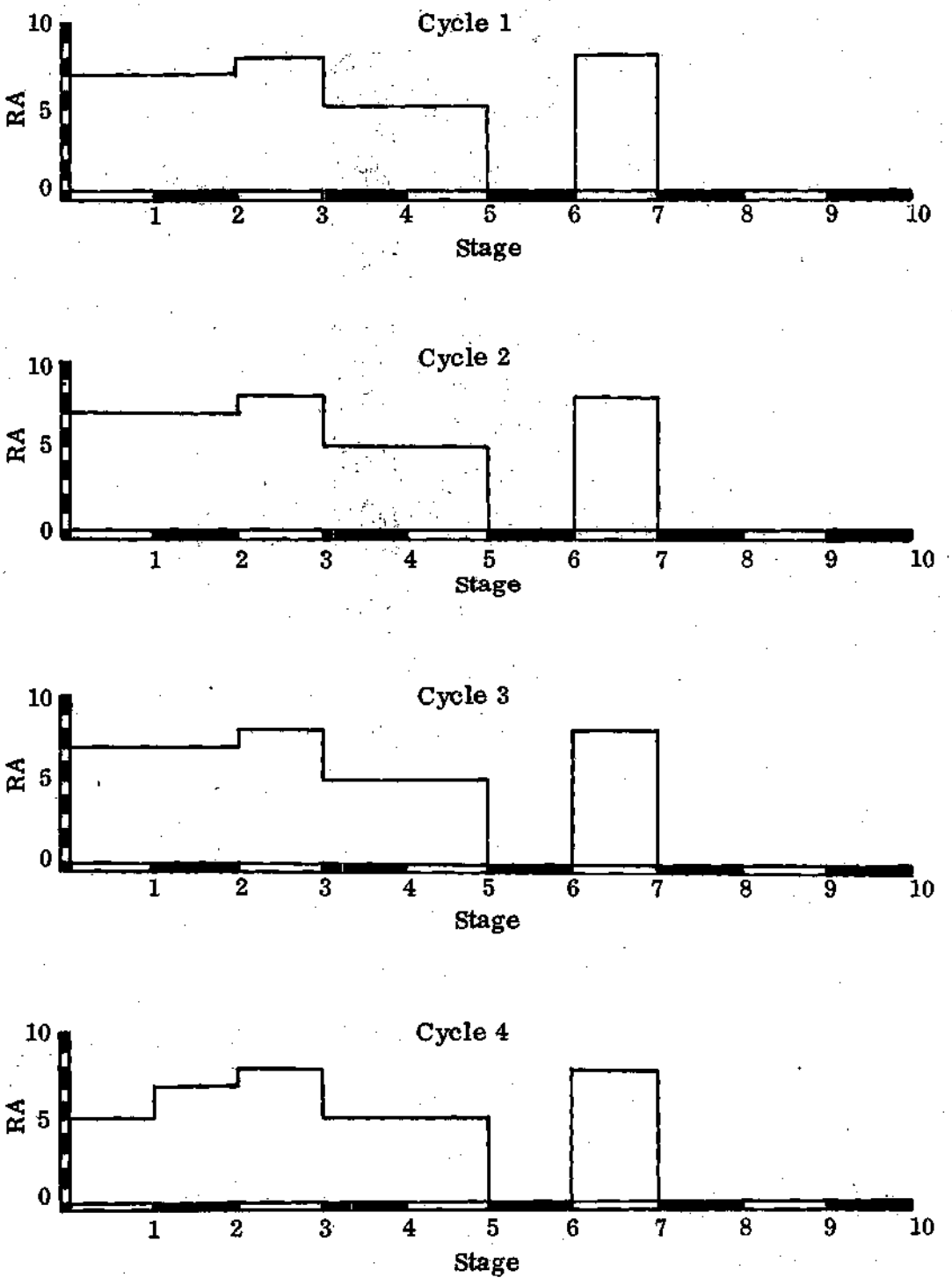


Figure 36. Resource Allocation for Resource Type Two, Iteration Two, Cycle One through Four.

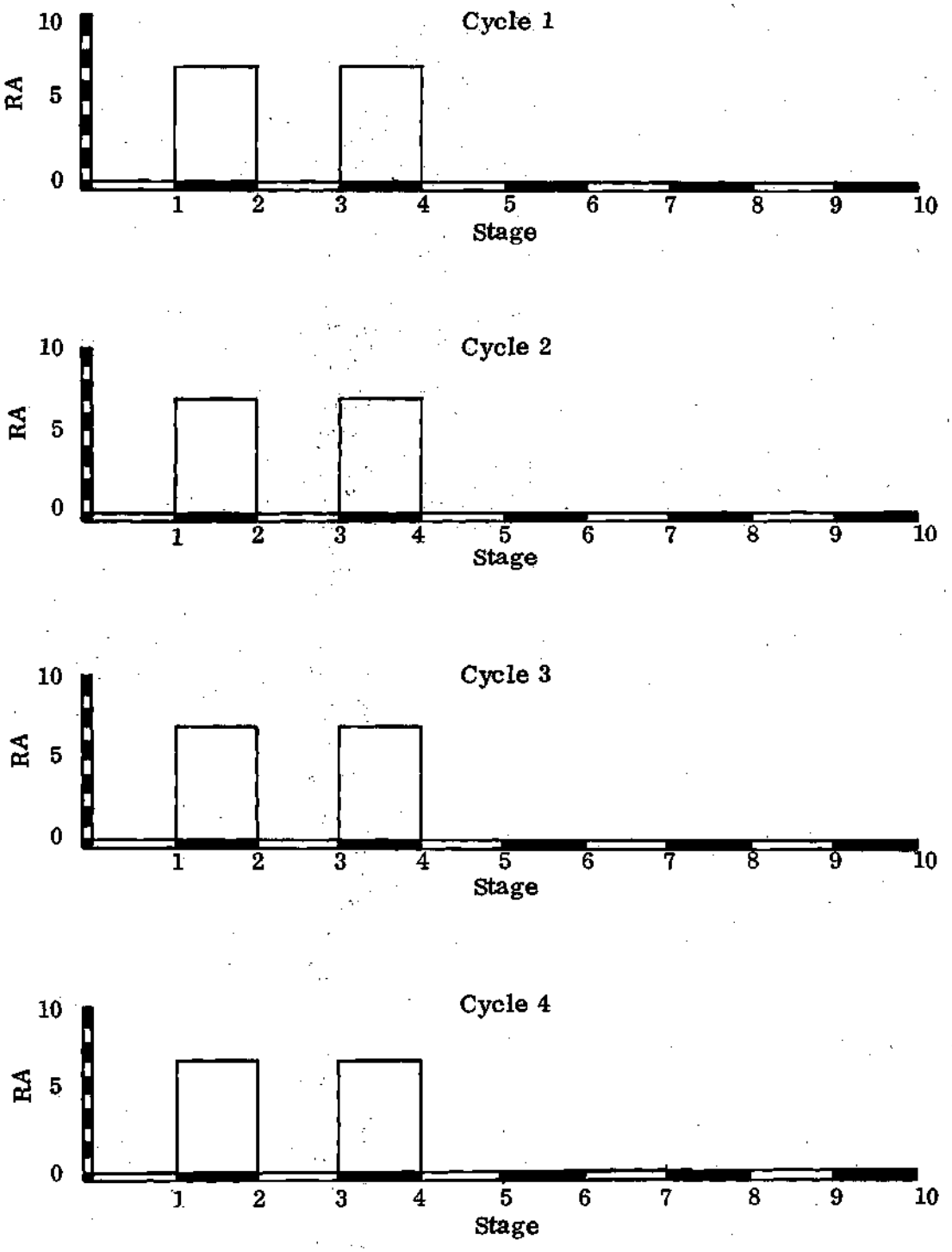


Figure 37. Resource Allocation for Resource Type Three, Iteration One, Cycles One through Four.

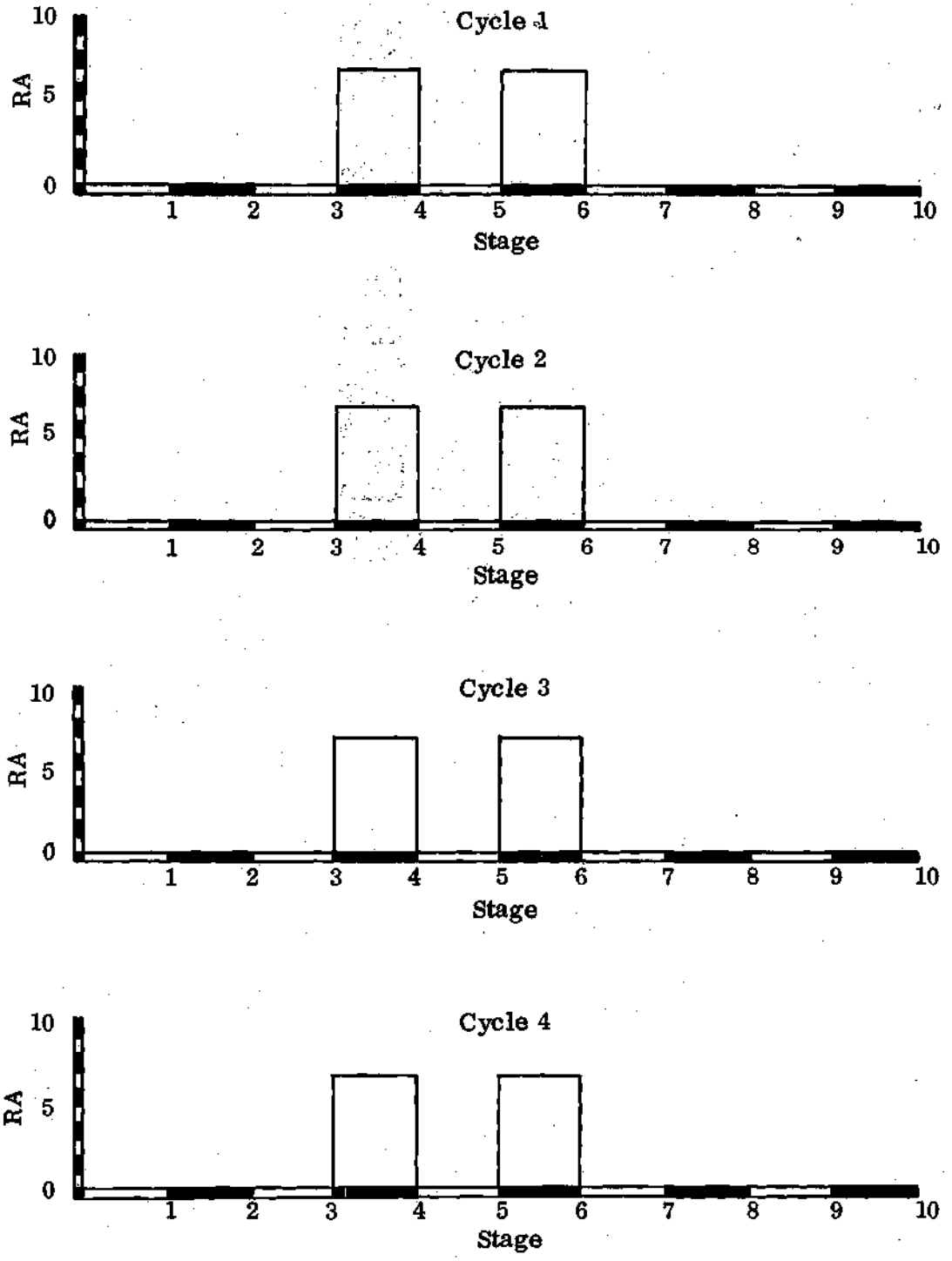


Figure 38. Resource Allocation for Resource Type Three, Iteration Two, Cycles One through Four.

little consequence except to test the alternative which is frequently seen in practice.

Figures 33 through 38 also show the final resource allocation resulting from the minimum cost allocation.

Minimum Time Schedules

The operation of the algorithm should allow generation of minimum time schedules by increasing the daily project cost or by increasing the idle cost to a point which will force maximum resource level assignments. Figure 39 shows the allocations resulting from applying a high daily project cost. The allocations are the maximum allowable and are just sufficient to accomplish the requirement. Figure 40 illustrates the network which results from the maximum allocations.

Alternately the minimum time schedule was tested by applying a very high penalty for idle resources. Figure 39 also shows results identical to the previous high project daily cost test.

The two tests could have produced different but accurate results. The high daily project cost alternative could have resulted in lower allocations, but the total project time should be the same. The high penalty for idle resources should always result in a minimum total project time with a maximum allocation level in each stage. The performance of the model on all input data was consistent in ability to produce a minimum project time.

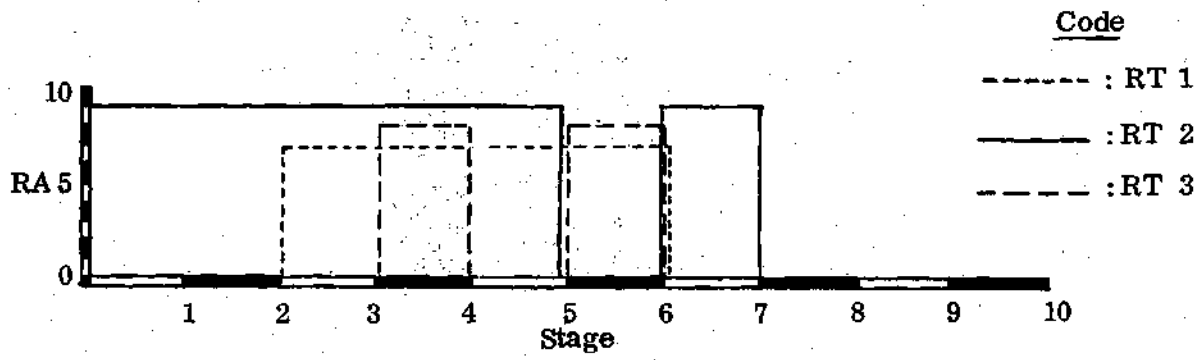


Figure 39A. Minimum Time from High Project Cost.

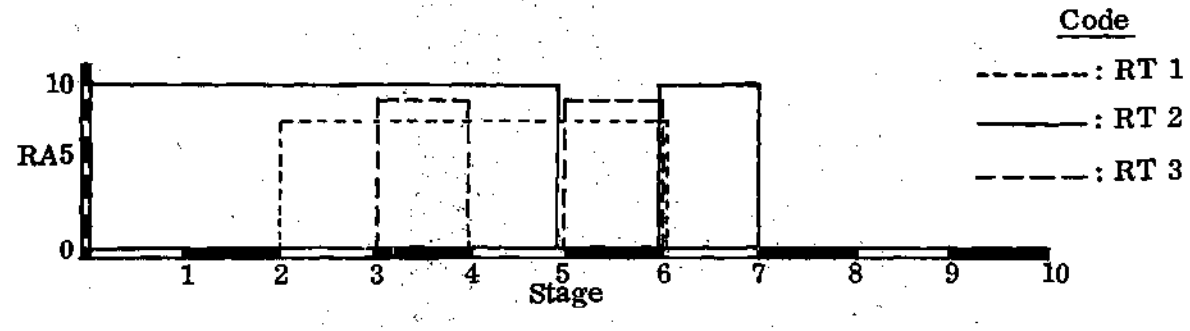


Figure 39B. Minimum Time from High Idle Cost.

Maximum Time Schedules

The maximum time schedule does not offer the same opportunity for performance as the minimum project time schedule. This is because the relaxation of the cost of project time still allows each resource type to search for its minimum cost level. Relaxation of both the cost of project time and cost of idle resources should however result in an allocation of the least costly resource level. This may not be the lowest level because two allocations of a low resource level may well be more costly than allocation of one high level which supplies as many resources as the two lower level allocations. Figure 40 shows that the model performed reasonably well on the example project data. Consistently

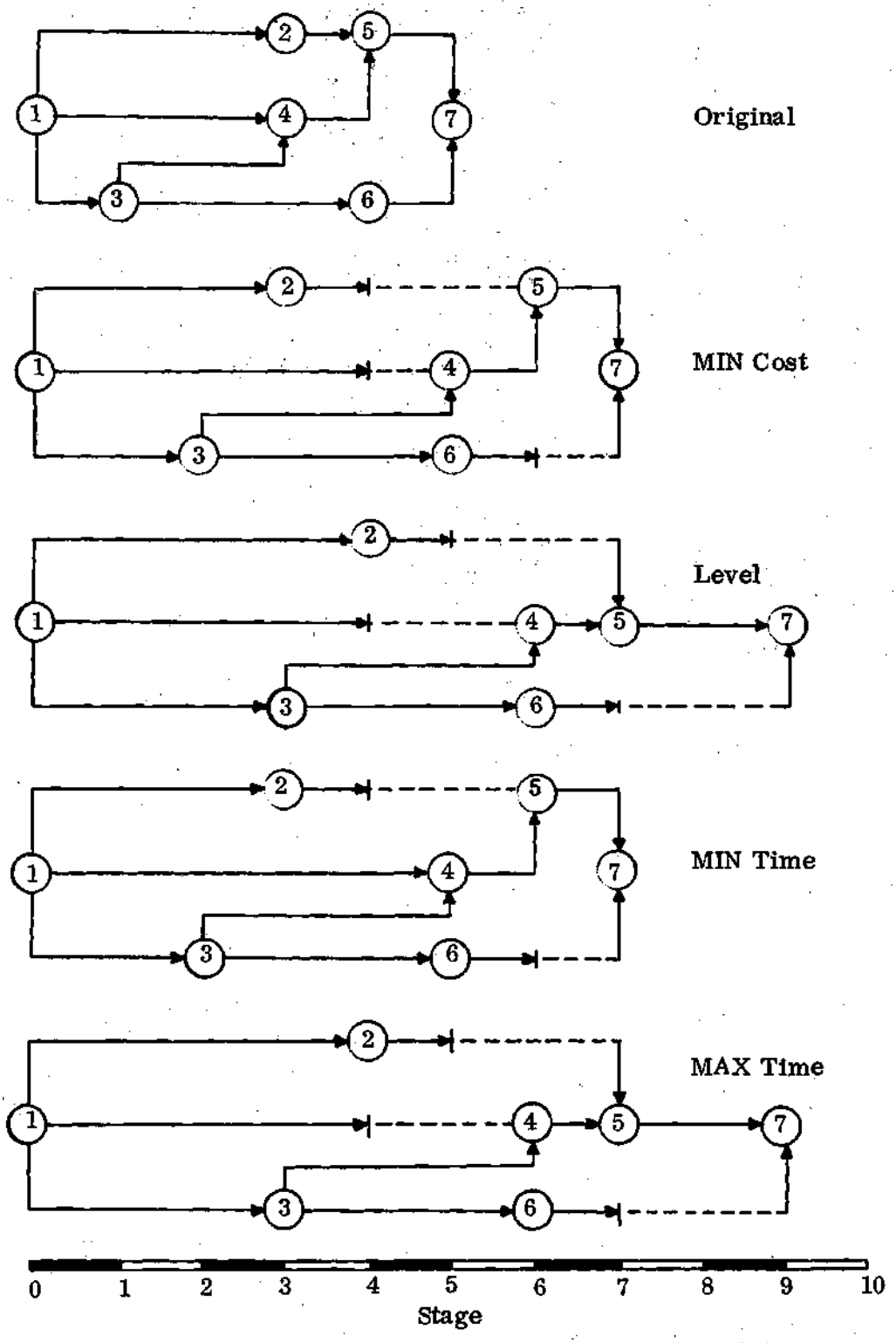


Figure 40. Time Scaled Networks for Test Options.

maximum schedules are not expected due to the sensitivity of the model to relative costs between resource levels.

Level Schedules

Level schedules were attempted by introducing a very high cost of changing resource levels. Figure 41 shows that the schedule tended towards level but was

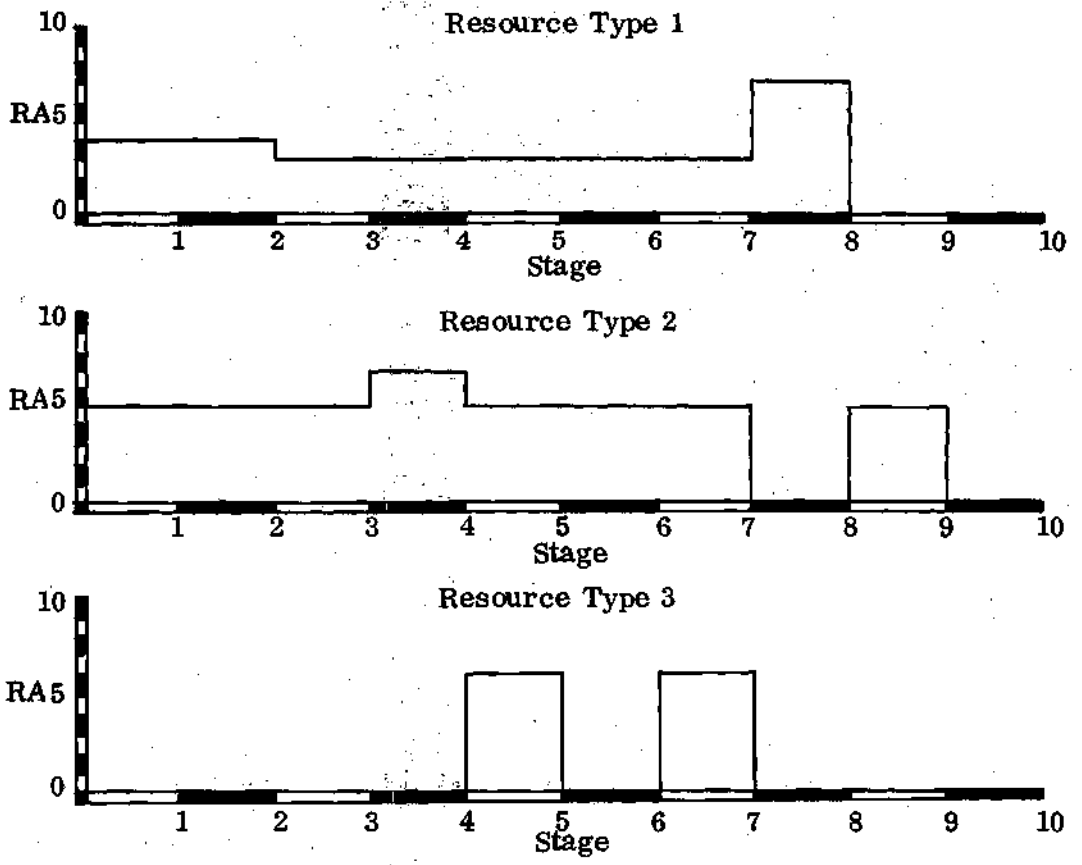


Figure 41. Resource Allocations for Level Schedules.

not level. Adequate performance was not expected due to the routine which allows a critical resource allocation to accumulate over time until the critical assignment is sufficiently in excess of the critical requirement to accomplish the total resource requirement. In addition, the level zero routine may follow an out of

context critical assignment and match an adjacent zero resource requirement with the high level instead of the level capable of producing an all level schedule.

Over-all performance of the algorithm for leveling was not considered adequate. The evaluation of the algorithm for minimum time schedules was considered better than the performance for minimum cost schedules. Adequacy for maximum time schedules was considered too data dependent to be appropriate and adequacy for level schedules was considered minimal.

Figure 42 is one of the larger networks which was tested for further evaluation of the model. The difficulty of verifying accurate performance on a network of this size was circumvented by structuring the input data to test for predictable results.

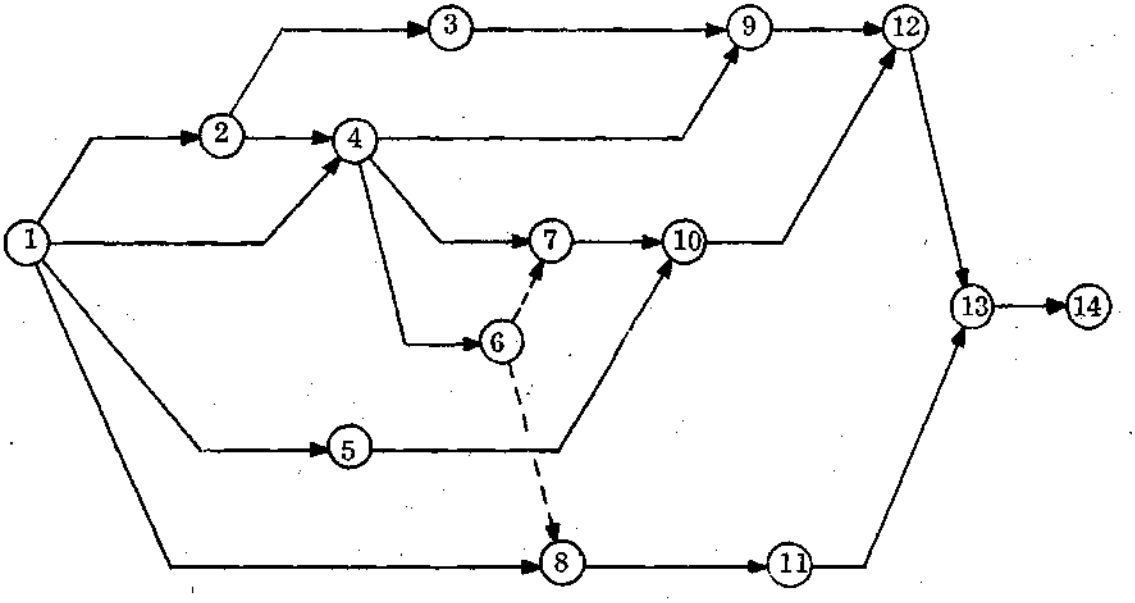


Figure 42. Example of Larger Network.

Table 15 is the data input for the network of Figure 42. As can be seen from the input data, resource type one allowed no alternative except assignment of one resource level and carried no restrictive change level cost or idle cost. The expected result is shown by Figure 43 and the resource assignments totaled 98 resource days as compared to the resource requirement of 91.

Resource type two was structured to test the selectivity between resource use costs and consequently the change level and idle costs were designed to have a minimal effect on the result. Figure 43 shows that the higher level of nine resources was selected in preference to the more costly level of six resources. In all 36 resource days were assigned to perform the 36 day resource requirement.

Resource type three was assigned costs which were equally expensive and not restricted by change level or idle costs. Figure 43 shows the assignments made to meet the resource requirements. The costs for resource type four were designed to force assignment of the lower level of eight resources. Figure 43 shows that the result did not select the lower level exclusively. This was caused by the resource requirement of stages 10 through 13 being larger than the minimum level.

Resource type five carried a definite cost advantage to the higher of the three alternative levels. Additionally, resource type five was restricted by a high change level and high idle cost. The result is shown by Figure 43 and was predictable.

Figure 44 is an example of the sub-network types evaluated for expected model performance. While these sub-networks do not represent all possible

configurations, the results appear sufficient to justify the conclusion that the model is operating in accordance with its design expectations.

Table 15. Input Data for Network of Figure 42

```
 1.  20066600000002.0
 2.  00100200102100700700700700701501501501501 5
 3.  00100400102100700700700700701501501501501 5
 4.  00100500302500600700800800803604204804804 8
 5.  00100800102800700700700700701501501501501 5
 6.  00200300101700700700700700701501501501501 5
 7.  00200400403400800900900900901503003003003 0
 8.  00300900402700800900900900901503003003003 0
 9.  00400600401800800900900900901503003003003 0
10.  00400700503000500700900900902102402902902 9
11.  00400900503900500700900900902102402902902 9
12.  00501000501500500700900900902102402902902 9
13.  00600700600000700700700700701501501501501 5
14.  00600800600000700700700700701501501501501 5
15.  00701000501500500700900900902102402902902 9
16.  00801100203600600900900900901602002002002 0
17.  00901200503500500700900900902102402902902 9
18.  01001200301400600700800800803604204804804 8
19.  01101300301200600700800800803604204804804 8
20.  01201300503100500700900900902102402902902 9
21.  01301400100400700700700700701501501501501 5
22.  0000001.000000001.0
23.  0000001.000000001.0
24.  0000001.000000001.0
25.  0000001.000000001.0
26.  0000043.000000015.0
27.  0000001.000000001.0
```

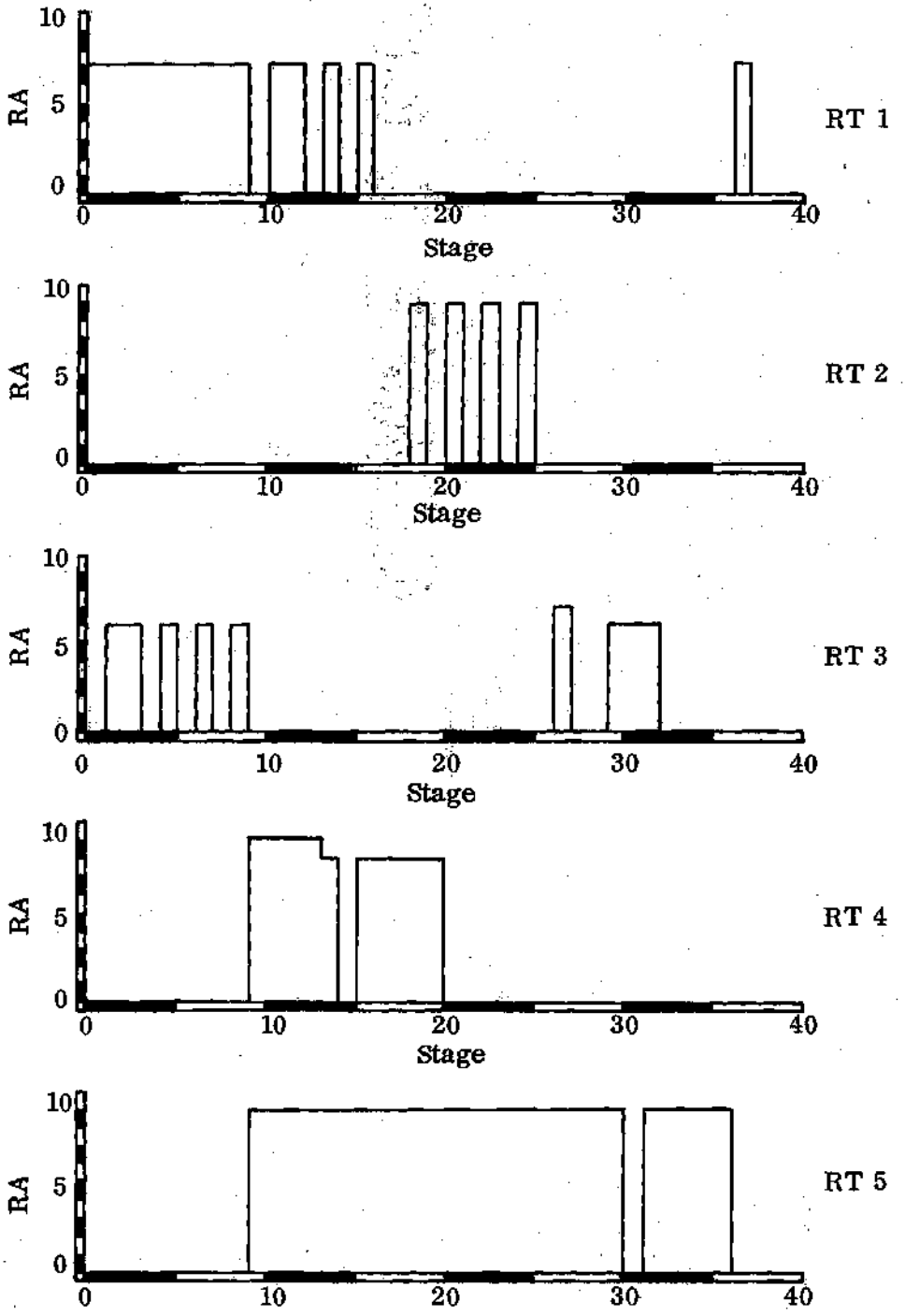


Figure 43. Resource Allocations for Figure 42.

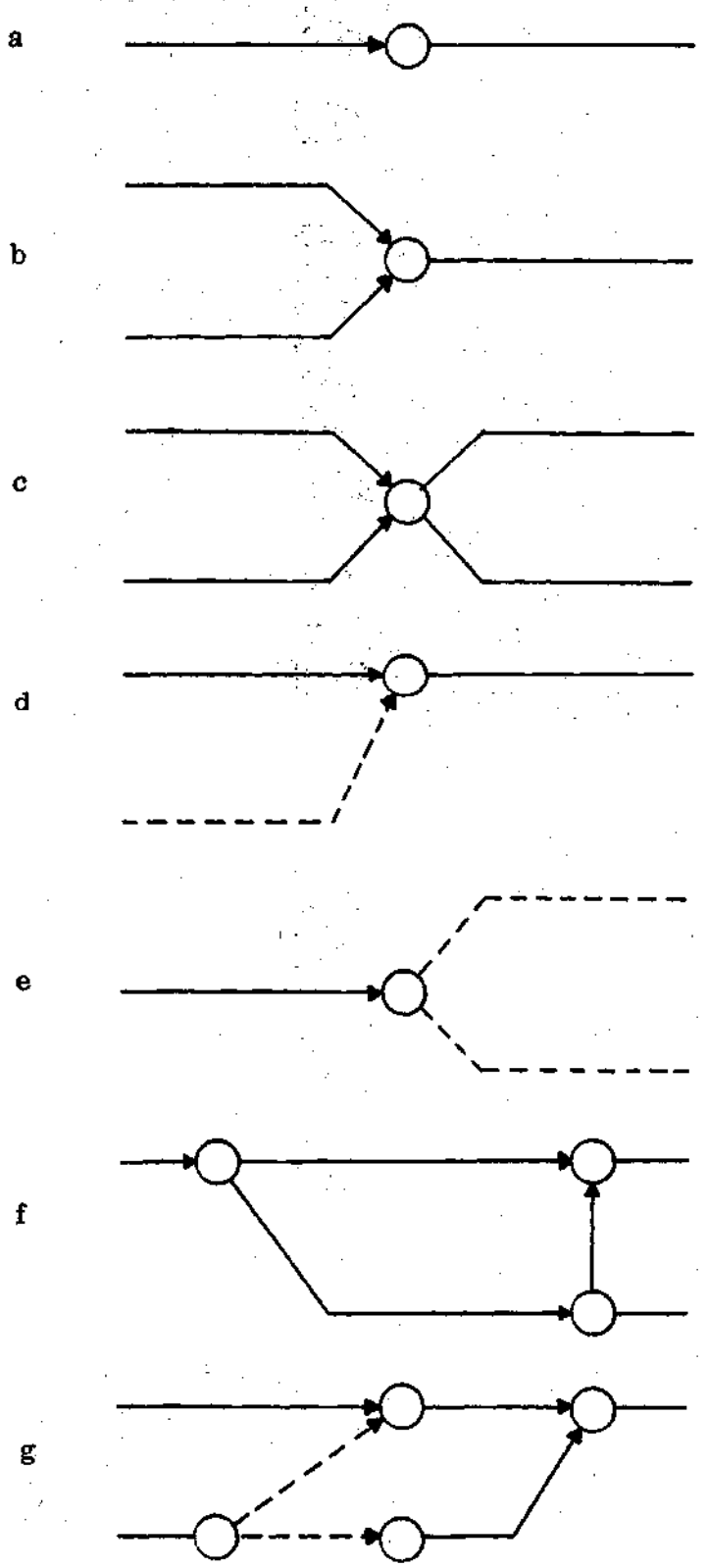


Figure 44. Subnetwork Types Tested for Algorithm Performance.

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

Research Contributions

The Resource Synthesis approach to the management of construction resources provides a methodology based upon optimal seeking procedures while retaining efficiency of solution for significant combinatorial problems. The results are obtained from a unique formulation of the constrained, multiple resource problem which employs the concept of a resource sensitive network which can be solved efficiently by a dynamic programming procedure developed especially for the construction resources problem.

The contributions of the methodology to the state-of-the-art in the construction environment are sixfold.

1. The methodology combines the effectiveness of leveling, time-cost trade-off and constrained resources allocation within a single approach which can be used without total comprehension of the complex algorithm. The synthesis of methodologies is accomplished by relating the cost of attaining each alternative as compared to the cost of other alternatives.
2. The Discrete Differential Dynamic Programming procedure is modified and adapted to the construction problem to improve the feasibility of the dynamic programming application by reducing the related machine storage requirements.

3. The concept of float in a critical path network is translated to a more useable quantity. Partitioned float is a portion of total float prorated to each activity on the basis of resource requirements. This concept alleviates the necessity of either the early and late state schedule and avoids extending non-critical activities to criticality by arbitrary processes. This procedure does not continuously maintain an all critical network and although this precludes contention of an optimal process the dispersion of actual activity durations as opposed to those estimated is considered to be at least as great as the remaining difference between critical and non-critical durations.
4. A procedure to advance the dynamic programming stage simultaneously with differing variables is developed. The method not only allows selection of minimum candidates which are feasible, but also carries alternatives which are infeasible in a current stage to future stages for consideration. This is accomplished by defining a one dimensional, indexed and feasible array which is used to locate minimum values for the traceback routine and transmitting minimum cost alternatives to future stages even though the alternative lacks feasibility in a current stage. The feasibility distinction among alternatives is aided by a two dimensional, sectional cost array which maintains separation of the critical and non-critical variables from which feasible minimums may be selected.
5. The utility of the CPM method has been extended to include a methodology which allows feasible time schedules in a practical, resource constrained environment. This extended capability can now provide useful information

to those responsible for its implementation and has the ability to facilitate a greater acceptance of the traditional CPM.

6. The concept of a resource schedule has been developed to augment the network precedence and contributes to the conceptualization of a project as the assignment of work to available resources over a definitive time period. This understanding departs from the activity precedence orientation and provides the framework for feasible time schedules within constrained resource situations.

Model Evaluation

The model developed is capable of scheduling ten resource types over a three working month time frame. The initial network input is not limited to the three month capability nor is the three month schedule the maximum limitation for allocation. The foreseeable future is recognized as a practical limitation due to the lack of reliable information at a date too far in the future. The arbitrary dimensioning of resource types to a maximum of ten provides a reasonable load for a scheduler's capability. However, as experience is gained in defining multiple resource levels, the desirability of working with an increased number of resource types may necessitate changing the dimensions of the type arrays for projects which can support the cost of using increased core storage.

The model proved compatible with complex networks which cannot be added either serially or in parallel as well as networks having activities common to multiple chains. Performance of the model was evaluated for all combinations

of no resource requirement, critical resource requirements and non critical resource requirements.

As presently dimensioned the model required approximately .041 hours of CPU time on an IBM 360 to compile and reach convergence for a twenty activity, six resource type, sixty-six day schedule. The cost of this small project schedule at current rates would be well within the budgetary allowance of a similar real project. Extension of the twenty activities would only result in a minor increase of costs since the major cost of core storage would not be significantly increased.

Recommendations for Further Study

The Resource Synthesis Program represents a prototype for a practical application of a resource allocation schedule in the construction field. As would be expected with a prototype model, several areas of further study have been identified in respect to the environment as well as the model.

1. As has been noted previously, there has been little valid research in the area of optimum resource mixture identification. Realizing that an optimum mixture is a dynamic variable which changes with each project, a stochastic probability function could be associated with mixture types. Such a function as an element of a data base would enhance the prediction of a valid performance ranges for a resource allocation schedule. With a more definitive knowledge of performance under varying conditions, the resource allocation schedule can also be changed to include stochastic variables. The result of

resource allocation could then produce an expected value in a probabilistic context.

2. The construction industry has been viewed as existing in a highly dynamic environment, where the unexpected has become the norm. Although the environment consists of a large number of variables, the variables themselves are identifiable; the impact, if any, is unknown. In such situations there exists a boundary, or perhaps multiple boundaries, beyond which the total construction project may be propelled into chaos. The similarity to catastrophe theory concepts may be adaptable as a means of improving mixture performance, identifying critical work and acceptable risk or even a rationalization for cost acceptance.
3. The psychological influence of resource mixtures and levels is related to the resource performance problem. The impact of this consideration may serve to invalidate performance data and provide a valuable link to safety criteria. The investigation of the psychological result of resource application could improve understanding of performance information and serve to better inform the construction industry of the nature of group phenomena in the industry.
4. The resource synthesis model does not differentiate between projects and although capable of handling multiple projects the incorporation of differing daily costs at alternate times should be an alternative for further research.
5. The resource synthesis program has a capability of sensitivity analysis by input data manipulation. However, further study should develop a complete

methodology to indicate sensitivity by machine capability. This extension would be beneficial for complex projects where the sensitive variables are less obvious.

6. The resource synthesis model is a prototype developed with the intention of proving the algorithm. As such, the efficiency of many parts of the code can be improved as well as performed better by such alterations as trading CPU time for storage and using dynamic arrays. Although core storage did not hamper computational feasibility, a maximum project limitation exists and further research with a more efficient code should be conducted to identify project sizes for different computer types of a current generation.
7. Finally, the algorithm is well within the capability of minicomputers. The cost of storage available from the computer should be investigated along with the benefit which could be exacted from the algorithm with a compatible dimension.

APPENDICES

APPENDIX A

SUB-ROUTINE CRIT

```
C*************************************************************************
C*************************************************************************
      SUBROUTINE CRIT(I,J,N,DIJ,ES,TF)
C
C        CPITICAL PATH PROGRAM
C
      DIMENSION I(50),J(50),DIJ(50),ES(50),TF(50)
      INTEGER DIJ,ES,TF
      INTEGER TI,TE,EF,FF
      DIMENSION TI(50),TE(50),EF(50),LS(50),LF(50)
     *,FF(50)
C
CCCCCCCC     N=TOTAL NUMBER OF JOBS
CCCCCCCC     I,J = ACTIVITY IJ ( I?J)
CCCCCCCC     DIJ = DURATION OF ACTIVITY IJ
CCCCCCCC     ES  = EARLY START
CCCCCCCC     LS  = LATE START
CCCCCCCC     EF  = EARLY FINISH
CCCCCCCC     LF  = LATE FINISH
CCCCCCCC     FF  = FREE FLOAT
CCCCCCCC     TF  = TOTAL FLOAT
C
      INDEX=1
      DO 10 K=1,N
      IF(I(K) .GE. J(K)) GO TO 997
      IF(I(K) .LT. INDEX) GO TO 998
      IF(I(K) .EQ. INDEX+1) INDEX=I(K)
   10 CONTINUE
      DO 20 K=1,N
      TI(K)=0
      TE(K)=9999
   20 CONTINUE
      DO 30 K=1,N
      MAX=TI(I(K))+DIJ(K)
      IF(TI(J(K)) .LT. MAX) TI(J(K))=MAX
   30 CONTINUE
C
C
      TE(J(N))=TI(J(N))
      DO 40 KK=1,N
      K=N+1-KK
      MIN=TE(J(K))-DIJ(K)
      IF(TE(I(K)) .GT. MIN) TE(I(K))=MIN
   40 CONTINUE
C
C
      DO 50 K=1,N
      ES(K)=TI(I(K))
      LS(K)=TE(J(K))-DIJ(K)
      EF(K)=TI(I(K))+DIJ(K)
      LF(K)=TE(J(K))
      TF(K)=TE(J(K))-TI(I(K))-DIJ(K)
      FF(K)=TI(J(K))-TI(I(K))-DIJ(K)
   50 CONTINUE
      WRITE(6,673)
  673 FORMAT(1H1,'   N   I   J   ES   LS   EF   LF   FF   TF'/)
      DO 676 K=1,N
      WRITE(6,674)K,I(K),J(K),ES(K),LS(K),EF(K),LF(K),FF(K),TF(K)
  674 FORMAT(9I5,2X/)
  676 CONTINUE
```

```
C
C
      RETURN
  997 WRITE(6,300)
  300  FORMAT('ERROR IN DATA,I?J')
      STOP
  998  WRITE(6,301)
  301  FORMAT(' ERROR IN DATA, #2')
      STOP
      END
```

APPENDIX B

SUB-ROUTINE CHAIN

```
C*************************************************************************
      SUBROUTINE CHAIN1(I,J,B,N,LAST,PF,TF,RRQD)
      DIMENSION I(50),J(50),B(50,66,2)
      DIMENSION LEN(80),CHAINS(80)
      INTEGER B,CURNOD,CHNUM,PF,TF
      LOGICAL FIRST,ANY
      DIMENSION PF(50),TF(50),RRQD(50)
      INTEGER WHICH
      DO 10 K=1,N
      IF (I(K) .NE. 1) GO TO 11
   10 CONTINUE
   11  NUMOF1=K-1
C
      DO 20 K=1,NUMOF1
      B(K,1,1)=I(K)
      B(K,1,2)=J(K)
      LEN(K)=1
C  20 CONTINUE
C
      CHNUM=NUMOF1
C  21   ANY=.TRUE.
C     DO 30 K=1,CHNUM
      K=1
  301 FIRST=.TRUE.
      CURNOD=B(K,LEN(K),2)
      IF (CURNOD .EQ. LAST) GO TO 30
      ANY=.FALSE.
      DO 40 L=1,N
      IF (I(L) .NE. CURNOD) GO TO 40
      IF(.NOT. FIRST) GO TO 35
      LEN(K)=LEN(K)+1
      B(K,LEN(K),1)=I(L)
      B(K,LEN(K),2)=J(L)
      FIRST=.FALSE.
      GO TO 40
   35 CHNUM=CHNUM+1
      LENKM1=LEN(K)-1
      DO 36 JJ=1,LENKM1
      B(CHNUM,JJ,1)=B(K,JJ,1)
      B(CHNUM,JJ,2)=B(K,JJ,2)
   36 CONTINUE
      LEN(CHNUM)=LEN(K)
      B(CHNUM,LEN(CHNUM),1)=I(L)
      B(CHNUM,LEN(CHNUM),2)=J(L)
   40 CONTINUE
C  30 K=K+1
      IF (K.LE.CHNUM) GO TO 301
      IF (.NOT.ANY) GO TO 21
   41 DO 45 K=1,50
      IF(B(K,1,1).EQ.0) GO TO 46
   45  CONTINUE
C46   NUMOFC=K
   46 NUMOFC=K-1
      DO60 K=1,N
      NN=0
      DO 55 NUM=1,NUMOFC
      LENUM=LEN(NUM)
      DO50 L=1,LENUM
      IF (B(NUM,L,1) .NE. I(K)) GO TO 50
      IF (B(NUM,L,2) .NE. J(K)) GO TO 50
```

```
      NN=NN+1
      CHAINS(NN)=NUM
      GO TO 55
   50 CONTINUE
   55 CONTINUE
      SMAX=0
      DO 70 L=1,NN
      SUM=0
      LENCHN=LEN(CHAINS(L))
      DO 80 M=1,LENCHN
      WHICH=IJ(A(CHAINS(L),M,1),B(CHAINS(L),M,2),I,J)
      SUM=SUM+RROD(WHICH)
   80 CONTINUE
      IF(SUM .GT. SMAX) SMAX=SUM
   70 CONTINUE
      PF(K)=RROD(K)/SMAX*TF(K)
   60 CONTINUE
      RETURN
      END
```

APPENDIX C

SUB-ROUTINE LOAD

```
      C*************************************************************************
            SUBROUTINE LOAD(I,J,NN,RRQD,TYPE,CRT,DIJ,ES,LOD,NNS)
            DIMENSION I(50),J(50),RRQD(50),TYPE(50),CRT(50),DIJ(50),ES(50)
            LOGICAL CRT,CRTT
            INTEGER TYPE,DIJ,ES
            REAL LOD(10,66,10)
            WRITE(6,200)
       200  FORMAT (' LOAD ARRAY '////)
            DO 1 K=1,10
            DO 1 KK=1,NNS
            DO 1 KKK=1,10
            LOD(K,KK,KKK)=0
          1  CONTINUE
      C
            MAX=0
            DO 5 K=1,NN
            IF (TYPE(K) .GT. MAX) MAX=TYPE(K)
          5  CONTINUE
      C
            DO 6 K=1,MAX
            DO 8 M=1,NN
            IF(TYPE(M) .NE. K) GO TO 8
            L1=ES(M)+1
            L2=ES(M)+DIJ(M)
            DO 9 N=L1,L2
            IF (CRT(M)) LOD(K,N,1)=LOD(K,N,1)+RRQD(M)/DIJ(M)
            IF (.NOT. CRT(M)) LOD(K,N,2)=LOD(K,N,2)+RRQD(M)/DIJ(M)
            IF (ES(M)+DIJ(M) .NE. N) GO TO 9
            LOD(K,N,3)=LOD(K,N,3)+1
            IADD=0
            IF (CRT(M)) IADD=10**7
           LOD(K,N,3+LOD(K,N,3))=(100*I(M)+J(M))*1000+RRQD(M)/DIJ(M)
          **+IADD
          9  CONTINUE
          8  CONTINUE
          6   CONTINUE
            RETURN
            END
```

APPENDIX D

SUB-ROUTINE PERMUT

```
      C*************************************************************************
            SUBROUTINE PERMUT(NS,RR,RL,CRTL,RID,CHLEVY,PC,COMB,CMIN,IST,JJJ)
      C
            INTEGER RL
            DIMENSION RL(6),CRL(6),CMIN(50),CRTL(6)
            DIMENSION COMB(50,6),IST(50),IPT(5)
            LOGICAL TEST
       2    FORMAT(I5,F7.3,5(F3.0,1X),5X,F7.2,I5)
       999  III=NS
            J=1
            I=1
            K=1
            L=1
            M=1
            INM=1
            INUM=1
            II=1
       3     I=J
            IF (II .EQ. 6) GO TO 1000
       4     IF (II .GT. 1) GO TO 20
       5    SRL=RL(I)
            IF (SRL .GE. RR) GO TO 13
       9    IF (I .EQ. 6) GO TO 15
       10   I=I+1
            GO TO 4
       13   IJMP=1
            GO TO 500
       14   GO TO 2
       15   II=II+1
            GO TO 3
       20   J=2
       25    IF( II.GT.2) GO TO 35
            SRL=RL(I)+RL(J)
            IF (SRL .GE. RR) GO TO 28
       27   J=J+1
            IF (J .GT. 6) GO TO 9
            GO TO 25
       28    IJMP=2
            GO TO 500
       30   IF (I.EQ. 6) GO TO 15
            GO TO 10
       33   IF (J .EQ. 2) GO TO 1000
            GO TO 9
       35   K=2
       40   IF(II .GT. 3) GO TO 50
            SRL=RL(I)+RL(J)+RL(K)
            IF (SRL .GE. RR) GO TO 43
            IF (K .EQ. 6) GO TO 45
       42   K=K+1
            GO TO 40
       43   IJMP=3
            GO TO 500
       44     IF (K .EQ. 2) GO TO 33
        45    IF (J .EQ. 6) GO TO 9
            GO TO 27
       50   L=2
            IF (II .GT. 4) GO TO 65
       55   SRL=RL(I)+RL(J)+RL(K)+RL(L)
            IF (SRL .GE. RR) GO TO 58
       57   L=L+1
```

```
      56    IF (L .GT. 6) GO TO 60
            GO TO 55
      58    IJMP=4
            GO TO 500
      59    IF(L .EQ. 2) GO TO 44
      60    IF (K.EQ.6) GO TO 45
            GO TO 42
      55    M=2
      70    SRL=RL(I)+RL(J)+RL(K)+RL(L)+RL(M)
            IF (SRL .GE. RR) GO TO 72
      71    M=M+1
            IF (M.GT.6) GO TO 56
            GO TO 70
      72    IJMP=5
            GO TO 500
       73       IF (M .EQ. 2) GO TO 59
            IF ( L .EQ. 6) GO TO 60
            GO TO 57
     500    CRL(1)=RL(I)
            IPT(1)=I
            CRL(2)=RL(J)
            IPT(2)=J
            CRL(3)=RL(K)
            IPT(3)=K
            CRL(4)=RL(L)
            IPT(4)=L
            CRL(5)=RL(M)
            IPT(5)=M
            TOT=0.0
            DO  1010 JJ=1,II
            CIDL=(RL(6)-CRL(JJ))*RID
            RUC=CRTL(IPT(JJ))
            RLDC=PC*FLOAT(JJ-1)
            SUBT=CIDL+RUC
     1010   TOT=TOT+SUBT
            TOT=TOT+RLDC
            CLC=0.0
            IIM=II-1
            IF (II.EQ.1) GO TO 1016
            DO 1015 JL=1,IIM
            JJP=JL+1
            PCLC=ABS((CRL(JJP)-CRL(JL))*CHLEVY)
     1015   CLC=CLC+PCLC
     1016   CST=TOT+CLC
            CMIN(INUM)=CST
            IF(INUM .EQ.1) GO TO 501
            TEST=.TRUE.
            INUM1=INUM-1
            DO 6 MM=1,INUM1,2
            IF(RL(I) .NE. COMB(MM,5)) GO TO 6
            LL=MM
            TEST=.FALSE.
        6    CONTINUE
            IF (TEST) GO TO 699
            IF(CMIN(INUM) .GT. CMIN(LL)) GO TO 1001
            INUM=INM
            CMIN(INUM)=CST
            GO TO 501
     699    INM=INUM
     501    COMB(INUM,5)=RL(I)
```

APPENDIX E

SUB-ROUTINE DP

```
      SUBROUTINE DP(NS,RT,COST,RA,ST,CHLEVY,RAB,STA,STB,STT,FFSTR,
     *INDEX,A,LOD,M,III,JJJ,SUMCR,RRQD,Z,RID,REL)
      COMMON ISAVE(66,10,6),ITER,ICYCLE
      DIMENSION III(50),JJJ(50),SUMCR(10),RRQD(50)
      REAL LOD(10,66,10)
      INTEGER RT,STT
      DIMENSION COST(6),RA(6,5),ST(6),REL(10,5)
      DIMENSION A(66,10,7,7),RAB(66,10,6,5)
      DIMENSION STA(66,10,6),STB(66,10,6),STT(66,10,6)
      DIMENSION INDEX(66,10,6),FFSTR(66,10,6),TST(6)
      LOGICAL TEST1
      INTEGER WHICH,ST
      LOGICAL TEST
      IF ((Z.EQ.1.0).AND.(NS.GT.1 )) GO TO 9011
      IF (NS .GT. 1) GO TO 121
      DO 888 I=1,3
      STA(NS,RT,I)=ST(I)
      STB(NS,RT,I)=ST(I)
      FFSTR(NS,RT,I)=COST(I)
      A(NS,RT,7,I)=COST(I)
      IF (Z.EQ.1.0) GO TO 888
      DO 10 K=1,5
      RAB(NS,RT,I,K)=RA(I,K)
   10 CONTINUE
  888 CONTINUE
      DO 885 I=4,6
      STA(NS,RT,I)=ST(I)
      STB(NS,RT,I)=ST(I)
      FFSTR(NS,RT,I)=COST(I)
      A(NS,RT,7,I)=COST(I)
      IF (Z.EQ.1.0) GO TO 885
      DO 11 K=1,5
      RAB(NS,RT,I,K)=RA(I,K)
   11  CONTINUE
  885 CONTINUE
      GO TO 7002
  121 DO 12 I=1,3
      STB(NS,RT,I)=ST(I)
      A(NS,RT,I,7)=COST(I)
      DO 12 K=1,5
      RAB(NS,RT,I,K)=RA(I,K)
   12 CONTINUE
      DO 13 I=4,6
      STB(NS,RT,I)=ST(I)
      A(NS,RT,I,7)=COST(I)
      DO 13 K=1,5
      RAB(NS,RT,I,K)=RA(I,K)
   13 CONTINUE
 9011 DO 20 I=1,6
      DO 20 J=1,6
      DO 19 II=1,5
      JJ=6-II
      IF (RAB(NS-1,RT,J,JJ) .NE. 0) GO TO 22
   19 CONTINUE
      JJ=5
   22 LAST=RAB(NS-1,RT,J,JJ)
      DO 18 II=1,5
      IF (RAB(NS,RT,I,II) .NE. 0) GO TO 24
   18 CONTINUE
      II=5
```

```
      24    FIRST=RAB(NS,RT,I,II)
            ADCOST=0.0
            IF(LAST .NE. FIRST) ADCOST=ABS(LAST-FIRST)*CHLEVY
            A(NS,RT,I,J)=A(NS,RT,I,7)+A(NS-1,RT,7,J)+ADCOST
      20    CONTINUE
      7002  SUMNC=0.0
            SUM1=0.0
            SUM2=0.0
            SUM3=0.0
            SUMCR(RT)=LOD(RT,NS,1)+SUMCR(RT)
            RR=LOD(RT,NS,2)
            SUMNC=SUMNC+RR
            SUMCR(RT)=SUMCR(RT)+SUMNC
      45    IF (NS.EQ.1) GO TO 4003
            DO 8079 J=1,6
            STT(NS,RT,J)=0.0
            INDEX(NS,RT,J)=1
            FFSTR(NS,RT,J)=A(NS,RT,J,1)
            TST(J)=STB(NS,RT,J)+STA(NS-1,RT,1)
            DO 30 I=2,3
            IF (FFSTR(NS,RT,J) .LE. A(NS,RT,J,I)) GO TO 30
            INDEX(NS,RT,J)=I
            FFSTR(NS,RT,J)=A(NS,RT,J,I)
            TST(J)=STB(NS,RT,J)+STA(NS-1,RT,I)
      30    CONTINUE
      8079  CONTINUE
            DO 90 J=1,3
            L=J+3
            DO 91 I=4,6
            TEST1=.FALSE.
            IDIV=I
            M1=I
            M2=M1
            DUMMY=A(NS,RT,J,I)
            DTEST=STB(NS ,RT,J)+STA(NS-1,RT,I)
            D=0.0
            SUM=0.0
            DO 55 KNS=2,NS
            INS=NS-KNS+2
            DO 56 MM=1,5
            SUM=SUM+RAB(INS-1,RT,M2,MM)
      56    CONTINUE
            IF (INS.EQ.2) GO TO 55
            M2=INDEX(INS-1,RT,M2)
      55    CONTINUE
            DO 74 MM=1,5
            D=D+RAB(NS,RT,J,MM)
      74    CONTINUE
            SUM=SUM+D
            IF (SUM.GE.SUMCR(RT)) GO TO 101
            TEST1=.TRUE.
      91    CONTINUE
      101   DO 95 I=1,6
            TEST=.FALSE.
            M1=I
            M2=M1
            D=0.0
            SUM=0.0
            DO 69 KNS=2,NS
            INS=NS-KNS+2
```

```
      DO 64 MM=1,5
      SUM=SUM+RAB(INS-1,RT,M2,MM)
64    CONTINUE
      IF (INS.EQ.2) GO TO 69
      M2=INDEX(INS-1,RT,M2)
 69   CONTINUE
      DO 65 MM=1,5
      D=D+RAB(NS,RT,L,MM)
65    CONTINUE
      SUM=SUM+D
      IF (SUM.GE.SUMCR(RT)) GO TO 109
      TEST=.TRUE.
95    CONTINUE
      IF (TEST) GO TO 106
109   IF (FFSTR(NS,RT,L).NE.A(NS,RT,L,I)) FFSTR(NS,RT,L)=A(NS,RT,L,I)
      IF (INDEX(NS,RT,L).NE.I) INDEX(NS,RT,L)=I
      IF (TST(L).NE.STB(NS,RT,L)+STA(NS-1,RT,I)) TST(L)=STB(NS,RT,L)+STA
     *(NS-1,RT,I)
      IF (DUMMY.GE.FFSTR(NS,RT,L)) GO TO 80
      IF (TEST1) GO TO 80
106   FFSTR(NS,RT,L)=DUMMY
      INDEX(NS,RT,L)=IDIN
      TST(L)=DTEST
      DO 300 LL=1,5
      PAR(NS,RT,L,LL)=RAB(NS,RT,J,LL)
300   CONTINUE
80    CONTINUE
      DO 39 J=1,6
      STT(NS,RT,J)=STT(NS,RT,J)+TST(J)
39    CONTINUE
      DO 40 J=1,6
      WRITE(6,200) NS,FFSTR(NS,RT,J),(RAB(NS,RT,J,I),I=1,5),
     *(RAB(NS-1,RT,INDEX(NS,RT,J),I),I=1,5),STT(NS,RT,J)
200   FORMAT(*STAGE=*,I2,F10.2,5F5.3,5X,5F5.3,5X,I5)
40    CONTINUE
      WRITE(6,999)
999   FORMAT(///)
      DO 50 J=1,6
      A(NS,RT,7,J)=FFSTR(NS,RT,J)
      STA(NS,RT,J)=STT(NS,RT,J)
50    CONTINUE
      GO TO 41
4003  DO 14 MM=1,5
      SUM1=RAB(1,RT,1,MM)+SUM1
      SUM2=RAB(1,RT,2,MM)+SUM2
      SUM3=RAB(1,RT,3,MM)+SUM3
14    CONTINUE
      GO TO 51
41    M1=1
      M2=2
      M3=3
      DO 6 MM=1,5
      SUM1=SUM1+RAB(NS,RT,1,MM)
      SUM2=SUM2+RAB(NS,RT,2,MM)
      SUM3=SUM3+RAB(NS,RT,3,MM)
6      CONTINUE
      DO 8  KNS=2, NS
      INS= NS-KNS+2
      M1=INDEX(INS,RT,M1)
      M2=INDEX(INS,RT,M2)
```

```
      M3=INDEX(INS,RT,M3)
      DO 9 MM=1,5
      SUM1=SUM1+RAB(INS-1,RT,M1,MM)
      SUM2=SUM2+RAB(INS-1,RT,M2,MM)
      SUM3=SUM3+RAB(INS-1,RT,M3,MM)
    9 CONTINUE
    8 CONTINUE
   51 IF(SUM1.LT.SUMCR(RT)) FFSTR(NS,RT,1)=FFSTR(NS,RT,1)
     **10**4
      IF(SUM2.LT.SUMCR(RT)) FFSTR(NS,RT,2)=FFSTR(NS,RT,2)
     **10**4
      IF (SUM3.LT.SUMCR(RT)) FFSTR(NS,RT,3)=FFSTR(NS,RT,3)
     **10**4
      WRITE(6,105)NS,RT,SUM1,SUM2,SUM3,SUMCR(RT),
     *SUMNC,(FFSTR(NS,RT,II),II=1,5)
  105 FORMAT(' STAGE =',I2,', RT =',I2,10F10.5)
      WRITE(6,5000) (FFSTR(NS,RT,NN),NN=4,6)
 5000 FORMAT(40X,3F10.3)
      RETURN
      END
```

```
      SUBROUTINE DDDP1(NS,RT,JJJ,COMB,IST,CMIN,ISEED,CCOST,RA,ST,RR,
     *SAVE1,SAVE2,SAVE3,SAVE4,SAVE5,SAVE6)
      INTEGER RT,ST
      COMMON ISAVE(66,10,5),ITER,ICYCLE
      DIMENSION COMB(50,6),CCOST(6),CMIN(50),IST(50)
      DIMENSION SAVE1(66,10,10,5),SAVE4(66,10,10,5),SAVE2(66,10,10)
      DIMENSION SAVE5(66,10,10),SAVE3(66,10,10),SAVE6(66,10,10)
      DIMENSION ST(6),RA(6,5)
      IF (ISEED .EQ.1) GO TO 404
      IF(RR .NE. 0) GO TO 16
      DO 15 K=1,3
      CCOST(K)=0
      ST(K)=0
      DO 18 KK=1,5
      RA(K,KK)=0
   18 CONTINUE
   15 CONTINUE
      ISAVE(NS,RT,1)=1
      ISAVE(NS,RT,2)=5
      ISAVE(NS,RT,3)=10
      DO 11 K=1,10
      SAVE3(NS,RT,K)=0.0
      SAVE2(NS,RT,K)=0.0
      DO 11 KK=1,5
      SAVE1(NS,RT,K,KK)=0.0
   11 CONTINUE
      RETURN
C     CRIT VALUES
C
C     SORT
C
   16 KK=1
   33 SMALL=10.0E+40
      J1=1
   44    IF (CMIN(J1) .LT. 0) GO TO 88
      IF (CMIN(J1) .GT. SMALL) GO TO 88
      K=J1
      SMALL=CMIN(J1)
   88 J1=J1+1
      IF (J1 .LE. JJJ) GO TO 44
      SAVE2(NS,RT,KK)=IST(K)
      SAVE3(NS,RT,KK)=CMIN(K)
      DO 99 M1=1,5
      SAVE1(NS,RT,KK,M1)=COMB(K,M1)
   99 CONTINUE
      CMIN(K)=-1
      KK=KK+1
      IF (KK .LE. JJJ) GO TO 33
C
C
      ISAVE(NS,RT,1)=1
      CCOST(1)=SAVE3(NS,RT,1)
      ST(1)=SAVE2(NS,RT,1)
      IMEAN=JJJ/2
      ISAVE(NS,RT,2)=IMEAN
      CCOST(2)=SAVE3(NS,RT,IMEAN)
      ST(2)=SAVE2(NS,RT,IMEAN)
      ISAVE(NS,RT,3)=JJJ
      CCOST(3)=SAVE3(NS,RT,JJJ)
```

```
      ST(3)=SAVE2(NS,RT,JJJ)
      DO 163 M1=1,5
      RA(1,M1)=SAVE1(NS,RT,1,M1)
      RA(2,M1)=SAVE1(NS,RT,IMEAN,M1)
      RA(3,M1)=SAVE1(NS,RT,JJJ,M1)
  163 CONTINUE
      GO TO 999
C
C   CRIT + NONCRIT VALUES
C   SORT
  404 IF (RR .NE. 0) GO TO 45
      DO 47 K=4,6
      ST(K)=0
      CCOST(K)=0
      DO 48 KK=1,5
      RA(K,KK)=0
   48  CONTINUE
   47  CONTINUE
      ISAVE(NS,RT,4)=1
      ISAVE(NS,RT,5)=3
      ISAVE(NS,RT,6)=5
      DO 23 K=1,10
      SAVE6(NS,RT,K)=0.0
      SAVE5(NS,RT,K)=0.0
      DO 23 KK=1,5
      SAVE4(NS,RT,K,KK)=0.0
   23 CONTINUE
      RETURN
   45  KK=1
  133  SMALL=10.0E+40
      J=1
  144 IF (CMIN(J) .LT. 0) GO TO 188
      IF (CMIN(J) .GT. SMALL) GO TO 188
      K=J
      SMALL=CMIN(J)
  188 J=J+1
      IF (J .LE. JJJ) GO TO 144
      SAVE5(NS,RT,KK)=IST(K)
      SAVE6(NS,RT,KK)=CMIN(K)
      DO 263 M1=1,5
      SAVE4(NS,RT,KK,M1)=COMB(K,M1)
  263 CONTINUE
      CMIN(K)=-1
      KK=KK+1
      IF (KK .LE. JJJ) GO TO 133
      ISAVE(NS,RT,4)=1
      CCOST(4)=SAVE6(NS,RT,1)
      ST(4)=SAVE5(NS,RT,1)
      IMEAN=JJJ/2
      ISAVE(NS,RT,5)=IMEAN
      CCOST(5)=SAVE6(NS,RT,IMEAN)
      ST(5)=SAVE5(NS,RT,IMEAN)
      ISAVE(NS,RT,6)=JJJ
      CCOST(6)=SAVE6(NS,RT,JJJ)
      ST(6)=SAVE5(NS,RT,JJJ)
      DO 353 M1=1,5
      RA(4,M1)=SAVE4(NS,RT,1,M1)
      RA(5,M1)=SAVE4(NS,RT,IMEAN,M1)
      RA(6,M1)=SAVE4(NS,RT,JJJ,M1)
  353 CONTINUE
```

```
      999  RETURN
           END
```

APPENDIX F

SUB-ROUTINE DDDP2

```
C*************************************************************************
      SUBROUTINE DODP2(NNS,FFSTR,OFSTR,OSTB,STB,M,INDEX,IL)
      COMMON ISAVE(66,10,6),ITER,ICYCLE
      DIMENSION IL(10)
      DIMENSION INDEX(66,10,6)
      DIMENSION FFSTR(66,10,6),OFSTR(10,66,10),OSTB(66,10)
      DIMENSION STB(66,10,6)
      INTEGER RT
C
      DO 16 RT=1,M
      IFPT=IL(RT)
      DO 15  KNS=1,NNS
      INS=NNS-KNS+1
      GO TO (171,50,60,70,80,90),IFPT
171   IDIF=ISAVE(INS,RT,2)-ISAVE(INS,RT,1)
      IF (ISAVE(INS,RT,1) .EQ. 1) GO TO 500
      IF (IDIF.EQ.1) ISAVE(INS,RT,2)=ISAVE(INS,RT,2)
      IF (IDIF.GE.3) ISAVE(INS,RT,2)=ISAVE(INS,RT,2)-2
      IF (IDIF.EQ.2) ISAVE(INS,RT,2)=ISAVE(INS,RT,2)-1
      ISAVE(INS,RT,3)=ISAVE(INS,RT,2)
      ISAVE(INS,RT,2)=ISAVE(INS,RT,1)
      ISAVE(INS,RT,1)=ISAVE(INS,RT,1)-1
      GO TO 36
 50   IDIF=ISAVE(INS,RT,2)-ISAVE(INS,RT,1)
      IF (IDIF.EQ.1) ISAVE(INS,RT,1)=ISAVE(INS,RT,1)
      IF(IDIF.GE.3) ISAVE(INS,RT,1)=ISAVE(INS,RT,1)+2
      IF (IDIF .EQ. 2) ISAVE(INS,RT,1)=ISAVE(INS,RT,1)+1
      IDIF=ISAVE(INS,RT,3)-ISAVE(INS,RT,2)
      IF (IDIF .EQ. 1) ISAVE(INS,RT,3)=ISAVE(INS,RT,3)
      IF(IDIF.GE.3) ISAVE(INS,RT,3)=ISAVE(INS,RT,3)-2
      IF (IDIF .EQ. 2)    ISAVE(INS,RT,3)=ISAVE(INS,RT,3)-1
      GO TO 36
 60   IDIF=ISAVE(INS,RT,3)-ISAVE(INS,RT,2)
      IF (ISAVE(INS,RT,3).EQ.10) GO TO 300
      IF(IDIF .EQ. 1) ISAVE(INS,RT,2)=ISAVE(INS,RT,2)
      IF (IDIF .GE.3) ISAVE(INS,RT,2)=ISAVE(INS,RT,2)-2
      IF (IDIF .EQ. 2)  ISAVE(INS,RT,2)=ISAVE(INS,RT,2)-1
      ISAVE(INS,RT,1)=ISAVE(INS,RT,2)
      ISAVE(INS,RT,2)=ISAVE(INS,RT,3)
      ISAVE(INS,RT,3)=ISAVE(INS,RT,3)+1
      GO TO 36
 70   IDIF=ISAVE(INS,RT,5)-ISAVE(INS,RT,4)
      IF (ISAVE(INS,RT,4) .EQ.1) GO TO 400
      IF (IDIF .EQ. 1) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)
      IF (IDIF.GE.3) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)-2
      IF (IDIF .EQ. 2) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)-1
      ISAVE(INS,RT,6)=ISAVE(INS,RT,5)
      ISAVE(INS,RT,5)=ISAVE(INS,RT,4)
      ISAVE(INS,RT,4)=ISAVE(INS,RT,4)-1
      GO TO 36
 80   IDIF=ISAVE(INS,RT,5)-ISAVE(INS,RT,4)
      IF (IDIF .EQ. 1) ISAVE(INS,RT,4)=ISAVE(INS,RT,4)
      IF (IDIF.GE.3) ISAVE(INS,RT,4)=ISAVE(INS,RT,4)+2
      IF (IDIF .EQ. 2) ISAVE(INS,RT,4)=ISAVE(INS,RT,4)+1
      IDIF=ISAVE(INS,RT,6)-ISAVE(INS,RT,5)
      IF (IDIF .EQ. 1) ISAVE(INS,RT,6)=ISAVE(INS,RT,6)
      IF (IDIF.GE.3) ISAVE(INS,RT,6)=ISAVE(INS,RT,6)-2
      IF (IDIF .EQ. 2)  ISAVE(INS,RT,6)=ISAVE(INS,RT,6)-1
      GO TO 36
 90   IDIF=ISAVE(INS,RT,6)-ISAVE(INS,RT,5)
```

```
      IF (ISAVE(INS,RT,6).EQ.10) GO TO 600
      IF (IDIF .EQ. 1) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)
      IF (IDIF.GE.3) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)-2
      IF (IDIF .EQ. 2) ISAVE(INS,RT,5)=ISAVE(INS,RT,5)-1
      ISAVE(INS,RT,4)=ISAVE(INS,RT,5)
      ISAVE(INS,RT,5)=ISAVE(INS,RT,6)
      ISAVE(INS,RT,6)=ISAVE(INS,RT,6)+1
      GO TO 36
  500 IF(ISAVE(INS,RT,2)-ISAVE(INS,RT,1).GT.1) ISAVE(INS,RT,2)=
     $ISAVE(INS,RT,2)-1
      IF(ISAVE(INS,RT,3)-ISAVE(INS,RT,2).GT.1) ISAVE(INS,RT,3)=
     *ISAVE(INS,RT,3)-1
      GO TO 36
  400 IF (ISAVE(INS,RT,5)-ISAVE(INS,RT,4).GT.1) ISAVE(INS,RT,5)=
     *ISAVE(INS,RT,5)-1
      IF (ISAVE(INS,RT,6)-ISAVE(INS,RT,5).GT.1) ISAVE(INS,RT,6)=
     *ISAVE(INS,RT,6)-1
      GO TO 36
  300 IF (ISAVE(INS,RT,3)-ISAVE(INS,RT,2).GT.1) ISAVE(INS,RT,2)=
     *ISAVE(INS,RT,2)+1
      IF(ISAVE(INS,RT,2)-ISAVE(INS,RT,1).GT.1) ISAVE(INS,RT,1)=
     *ISAVE(INS,RT,1)+1
      GO TO 36
  600 IF (ISAVE(INS,RT,6)-ISAVE(INS,RT,5).GT.1) ISAVE(INS,RT,5)=
     *ISAVE(INS,RT,5)+1
      IF(ISAVE(INS,RT,5)-ISAVE(INS,RT,4).GT.1) ISAVE(INS,RT,4)=
     *ISAVE(INS,RT,4)+1
   36 OSTR(INS,RT)=STR(INS,RT,IFPT)
      IF (INS.EQ.1) GO TO 16
      IFPT=INDEX(INS,RT,IFPT)
   15 CONTINUE
   16 CONTINUE
C
      RETURN
      END
```

APPENDIX G

SUB-ROUTINE DURMOD

```
C*************************************************************************
      SUBROUTINE DURMOD(NNS,N,ES,DIJ,M,OSTB,TYPE)
      INTEGER ES,DIJ,TYPE
      COMMON ISAVE(66,10,6), ITER,ICYCLE
      DIMENSION ES(50),DIJ(50)
      DIMENSION  OSTB(66,10)
      DIMENSION TYPE(50)
      INTEGER RT
C
      DO 26 INS=1,NNS
      DO 25 MM=1,N
      IF ((ES(MM)+1 .LE.INS).AND.(ES(MM)+DIJ(MM).GE.INS)) GO TO 20
      GO TO 25
 20   RT=TYPE(MM)
      DIJ(MM)=DIJ(MM)+OSTB(INS,RT)-1
      IF (DIJ(MM).EQ.0) DIJ(MM)=1
 25    CONTINUE
 26   CONTINUE
      ISUM=0
      DO 27 INS=1,NNS
      MAX=0
      DO 28 K=1,M
      IF (OSTB(INS,K) .GT. MAX) MAX=OSTB(INS,K)
 28   CONTINUE
      ISUM=ISUM+MAX
  27  CONTINUE
      NNS=ISUM
      IF (NNS.GT.66) NNS=66
      RETURN
      END
```

APPENDIX H

MAIN PROGRAM

```
      *// THE NUMBER OF STAGES IS *,I3,2X,/
      *' THE DAILY COST IS',2X,F7.2,/////)
       WRITE(6,201)
  201  FORMAT(' THE INPUT DATA IS'///,4X,'N   I   J TYPE RRQD--
     * RLEV-----',
     *2X,'COST-----              CHLEV     IDLE      DUR')
       ITER=0
       DO 5 K=1,N
       READ(5,101) I(K),J(K),TYPE(K),RRQD(K),(RLEV(K,KK),KK=1,5),
     *  (COST(K,KK),KK=1,5)
  101  FORMAT(3I3,F3.0,10I3)
       IF(RRQD(K) .NE. 0) GO TO 7357
       TYPE(K)=0
       RLEV(K,KK)=0
       COST(K,KK)=0
       DIJ(K)=0
       GO TO 7358
 7357  MAXR=RLEV(K,1)
       IF (MAXR .LT. RLEV(K,2)) MAXR=RLEV(K,2)
       IF (MAXR .LT. RLEV(K,3)) MAXR=RLEV(K,3)
         IF (MAXR .LT. RLEV(K,4)) MAXR=RLEV(K,4)
       IF (MAXR .LT. RLEV(K,5)) MAXR=RLEV(K,5)
       DIJ(K)=RRQD(K)/MAXR
       IF (DIJ(K) .EQ.0) DIJ(K)=1
       IF (RRQD(K)/DIJ(K) .NE. MAXR) DIJ(K)=RRQD(K)/MAXR+1
 7358  WRITE(6,202)
     1  K,I(K),J(K),TYPE(K),RRQD(K),(RLEV(K,KK),KK=1,5),
     *   (COST(K,KK),KK=1,5),DIJ(K)
  202  FORMAT(/,1H ,4I4,F7.3,11I4)
    5  CONTINUE
       DO 99 K=1,M
       READ(5,220) CHLEV(K),IDLE(K)
  220  FORMAT(2F10.2)
   99  CONTINUE
       DO 6 K=1,M
       DO 8 KK=1,N
       IF (TYPE(KK) .NE. K) GO TO 8
       DO 9 KKK=1,5
       RELL(K,KKK)=RLEV(KK,KKK)
       COSTL(K,KKK)=COST(KK,KKK)
    9  CONTINUE
    8  CONTINUE
    6   CONTINUE
  400  ITER=ITER+1
       IF (ITER.GT.10) GO TO 999
       DO 401  RT=1,M
       CFSTR(ITER,RT)=0.0
  401  CONTINUE
       CALL CRIT(I,J,N,DIJ,ES,TF)
       CALL CHAIN1(I,J,B,N,J(N),PF,TF,RRQD)
       DO 7 K=1,N
       DUR(K)=DIJ(K)
       DIJ(K)=DIJ(K)+PF(K)
    7  CONTINUE
       CALL CRIT(I,J,N,DIJ,ES,TF)
       IF (ITER .NE. 1) GO TO 16
   16  DO 24 K=1,N
       IF( TF(K).EQ. 0) CRT(K)=.TRUE.
       IF( TF(K) .NE. 0) CRT(K)=.FALSE.
   24  CONTINUE
```

```
      CALL LOAD(I,J,N,RRQD,TYPE,CRT,DIJ,ES,LOD,NNS)
      ICYCLE=0
  500 ICYCLE=ICYCLE+1
      IF(ICYCLE.GT.4) GO TO 651
      DO 39 RT=1,M
      SUMCR(RT)=0.0
   39 CONTINUE
      IF(ICYCLE .GT. 1 ) GO TO 600
      Z=0.0
      DO 668 NS=1,NNS
      DO 669 RT=1,M
      RR=LOD(RT,NS,1)
      ISEED=0
      RL(1)=0
      CRTL(1)=0.0
      DO 13 LL=2,6
      RL(LL)=REL(RT,LL-1)
      CRTL(LL)=COSTL(RT,LL-1)
   13 CONTINUE
      RID=IDLE(RT)
      CHLEVY=CHLEV(RT)
      CALL PERMUT(NS,RR,RL,CRTL,RID,CHLEVY,DAILYC,COMB,CMIN,IST,JJJ)
      CALL DODP1(NS,RT,JJJ,COMB,IST,CMIN,ISEED,CCOST,RA,ST,RR,
     *SAVE1,SAVE2,SAVE3,SAVE4,SAVE5,SAVE6)
      RR=LOD(RT,NS,1)+LOD(RT,NS,2)
      ISEED=1
      CALL PERMUT(NS,RR,RL,CRTL,RID,CHLEVY,DAILYC,COMB,CMIN,IST,JJJ)
      CALL DODP1(NS,RT,JJJ,COMB,IST,CMIN,ISEED,CCOST,RA,ST,RR,
     *SAVE1,SAVE2,SAVE3,SAVE4,SAVE5,SAVE6)
      CALL DP(NS,RT,CCOST,RA,ST,CHLEVY,RAB,STA,STB,STT,FFSTR,
     *INDEX,A,LOD,M,I,J,SUMCR,RRQD,Z)
  669 CONTINUE
  668 CONTINUE
  600 DO 27 RT=1,M
      OFSTR(ICYCLE,NNS,RT)=FFSTR(NNS,RT,1)
      OSTB(NNS,RT)=STB(NNS,RT,1)
      IFPT=1
      DO 17 JJ=2,6
      IF (FFSTR(NNS,RT,JJ) .GT. OFSTR(ICYCLE,NNS,RT)) GO TO 17
      OFSTR(ICYCLE,NNS,RT)=FFSTR(NNS,RT,JJ)
      OSTB(NNS,RT)=STB(NNS,RT,JJ)
      IFPT=JJ
   17 CONTINUE
      IL(RT)=IFPT
   27 CONTINUE
      IF (ICYCLE.EQ.1) GO TO 500
      Z=1.0
      CALL DODP2(NNS,FFSTR,OFSTR,OSTB,STB,M,INDEX,IL)
      DO 117 NS=1,NNS
      DO 118 RT=1,M
      DO 19 L=1,3
      CCOST(L)=SAVE3(NS,RT,ISAVE(NS,RT,L))
      ST(L)=SAVE2(NS,RT,ISAVE(NS,RT,L))
      DO 19 LL=1,5
      RAB(NS,RT,L,LL)=SAVE1(NS,RT,ISAVE(NS,RT,L),LL)
   19 CONTINUE
      DO 20 L=4,6
      CCOST(L)=SAVE6(NS,RT,ISAVE(NS,RT,L))
      ST(L)=SAVE5(NS,RT,ISAVE(NS,RT,L))
      DO 20 LL=1,5
```

```
      RAB(NS,RT,L,LL)=SAVE4(NS,RT,ISAVE(NS,RT,L),LL)
20    CONTINUE
      CALL DF(NS,RT,CCOST,RA,ST,CHLEVY,RAB,STA,STB,STT,FFSTR,
     *INDEX,A,LOD,M,I,J,SUMCR,RROD,Z,IDLE,REL)
118   CONTINUE
119   CONTINUE
117   CONTINUE
      DO 15 RT=1,M
      CFSTR(ITER,RT)=CFSTR(ICYCLE,NNS,RT)+CFSTR(ITER,RT)
      WRITE(6,104) ICYCLE,ITER,CFSTR(ITER,RT)
104   FORMAT(' CYCLE=',I2,', ITER=',I2,F10.5)
15    CONTINUE
      GO TO 500
651   IF (ITER.EQ.1) GO TO 1400
      TEST=.FALSE.
      DO 1500 RT=1,M
      IF (CFSTR(ITER,RT) .GE. (.99*CFSTR(ITER-1,RT))) TEST=.TRUE.
1500  CONTINUE
      IF (TEST) GO TO 999
1400  CALL OURMOD(NNS,N,ES,DIJ,M,OSTB,TYPE)
      GO TO 400
999   DO 2007 RT=1,M
      WRITE(6,2000) RT
2000  FORMAT(1H1,54X,'RESOURCE',1X,I2,//5X,'STAGE',10X,'I',5X,
     *'J',10X,'CRR',15X,'TRR',15X,'RA')
      IF=IL(RT)
      DO 2006 KNS=1,NNS
      K=NNS-KNS+1
      DO 2003 ML=1,N
      IF(TYPE(ML) .EQ. RT) GO TO 2001
      GO TO 2003
2001  IF((ES(ML)+1 .LE. K) .AND. (ES(ML)+DIJ(ML) .GE. K))
     *GO TO 2050
      GO TO 2003
2050  WRITE(6,2004)(RT,ES(ML)+DIJ(ML),3)
      DO 2005 KK=1,NN
      JYY=LOD(RT,ES(ML)+DIJ(ML),3+KK)
      JJ=MOD(JYY/1000,100)
      II=MOD(JYY/100000,100)
      IF (LOD(RT,ES(ML)+DIJ(ML),3+KK) .LT. 10**5 ) GO TO 2013
      X=RROD(ML)/FLOAT(DIJ(ML))
      XM=Y
      GO TO 6331
2013  XM=RROD(ML)/FLOAT(DIJ(ML))
      X=0.0
      IF(KNS .EQ. 1) GO TO 2011
6331  IF(KNS .EQ. NNS) GO TO 2011
      DO 6328 JK=2,4
      DO 6328 JJL=2,5
      RABBB(KNS,RT)=RAB(KNS,RT,1,1)+RAB(KNS,RT,JK,JJL)
6328  CONTINUE
      IF(RABBB(KNS,RT) .NE. 0) GO TO 2011
      IIL(RT)=IFPT
      IIF=IIL(RT)
      IK=NNS-KNS+1
      IIF=INDEX(IK,RT,IIF)
      DO 6309 IS=1,5
      IF(RAB(IK-1,RT,IIF,IS) .NE. 0) GO TO 6310
6309  CONTINUE
      IS=5
```

```
      6310 LAST=RAB(IK-1,RT,IIF,IS)
           DO 6312 ISS=1,5
           IF(RAB(IK+1,RT,IIF,ISS) .NE. 0) GO TO 6311
      6312 CONTINUE
           ISS=5
      6311 NEXT=RAB(IK+1,RT,IIF,ISS)
           NEXTZ=NEXT*CHLEVY
           IF(NEXT .GT. LAST) GO TO 6321
           NEXTL=(LAST-NEXT)*CHLEVY
           GO TO 6322
      6321 NEXTL=(NEXT-LAST)*CHLEVY
      6322 DO 6315 JR=1,5
           IF(REL(RT,JR) .NE. LAST) GO TO 6315
           LEVEL=COSTL(RT,JR)+(REL(RT,5)-REL(RT,JR))*RID*NEXTL
           ZERO=NEXTZ+LAST*CHLEVY+(REL(RT,5)*RID)
      6315 CONTINUE
           IF(LEVEL .GT. ZERO) GO TO 6314
           RAB(IK,RT,IIF,5)=LAST
           STR(IK,RT,IIF)=1
           FFSTR(IK,RT,IIF)=LEVEL
      6314 RAB(IK,RT,IIF,5)=0
           STR(IK,RT,IIF)=1
           FFSTR(IK,RT,IIF)=ZERO
      2011 IF ((K.EQ.NNS).AND.(KK.EQ.1)) WRITE(6,2008)K,II,JJ,X,XM,
          *(RAB(K,RT,IF,LL),LL=1,5)
           IF((K.EQ.NNS).AND.(KK.NE.1)) WRITE(6,2009) II,JJ,X,XM,
          *(RAB(K,RT,IF,LL),LL=1,5)
           IF ((K.NE.NNS).AND.(KK.EQ.1)) WRITE(6,2008)K,II,JJ,X,XM,
          $(RAB(K,RT,IF,LL),LL=1,5)
      2008 FORMAT(7X,I2,10X,I2,4X,I2,10X,F3.1,15X,F3.1,15X,5F5.2)
           IF ((K.NE.NNS).AND.(KK.NE.1)) WRITE(6,2009)II,JJ,X,XM,
          1(RAB(K,RT,IF,LL),LL=1,5)
      2009 FORMAT(19X,I2,4X,I2,10X,F3.1,15X,F3.1,15X,5F5.2)
      2005 CONTINUE
      2003 CONTINUE
           IF (K.EQ.1) GO TO 2006
           IF=INDEX(K,RT,IF)
      2006 CONTINUE
      2007 CONTINUE
           ICYCLE=10
           DO 4000 KK=1,80
           DO 4000 KKK=1,80
           GRAPH1(KK,KKK)=BLANQ
      4000 CONTINUE
           OMAX=OFSTR(1,NNS,1)
           DO 4005 RT=1,M
           DO 4005 IC=1,ICYCLE
           IF(OMAX.LT.OFSTR(IC,NNS,RT)) OMAX=OFSTR(IC,NNS,RT)
      4005 CONTINUE
           DO 4006 IC=1,ICYCLE
           DO 4006 RT=1,M
           LL=(OFSTR(IC,NNS,RT)/OMAX)*80.0
           GRAPH1(LL,IC*8)=NUMS(RT)
      4006 CONTINUE
           DO 4051 JJ=1,80
           JM1=80-JJ+1
           WRITE(6,4050) (GRAPH1(JM1,J3),J3=1,80)
      4050 FORMAT(1X,'.',80A1)
      4051 CONTINUE
           WRITE(6,4060)
```

```
      4060 FORMAT(2X,'....1.........2.........3.........4.........5.........6
     *.........7.........8.........9.........10....',//,40X,'CYCLES')
           STOP
           END
```

```
      FUNCTION IJ(II,JJ,I,J)
      DIMENSION I(50),J(50)
      DO 10 K=1,50
      IF (I(K) .NE. II .OR. J(K) .NE. JJ) GO TO 10
      IJ=K
      RETURN
   10 CONTINUE
      STOP
      END
```

LITERATURE CITED

1. Antill, James M. and R. W. Woodhead, Critical Path Methods in Construction Practices, 2nd Edition, John Wiley & Sons, Inc., 1970.

2. Ashley, William F. and M. T. Austin, "Case Studies in Network Planning, Scheduling and Control of Research and Development Projects," Chapter 12, Operations Research in Research and Development, Edited by Burton V. Dean, John Wiley & Sons, Inc., 1963.

3. Baker, James J. and L. R. Shaffer, "Staged Decision Theory Applied to the Limited Resource Problem," Construction Research Series No. 8, Department of Civil Engineering, University of Illinois, September 1965.

4. Balas, E., "An Additive Algorithm for Solving Linear Programs with 0-1 Variables," Operations Research 13, 517-549 (1965).

5. Balas, E. and R. E. Small, "Mixed Integer Programming by a Branch and Bound Technique," paper presented at IFIP Congress 1965, New York, May 1965.

6. Balas, E., "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," Oper. Res. 17, 941-957 (1969).

7. Balas, E., "Project Scheduling with Resource Constraints," Proc. NATO Conference on Applications of Mathematical Programming, June 24-28, 1968, Cambridge, Eng., American Elsevier, N. Y., 1970.

8. Battersby, Albert, Network Analysis for Planning and Scheduling, John Wiley and Sons, 1970.

9. Bellman, R., Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.

10. Bennett, F. Lawrence, "Critical Path Resource Scheduling Algorithm," Journal of the Construction Division, ASCE Proceedings, Vol. 94, No. C02, (October, 1968), pp. 161-180.

11. Benson, L. A. and R. F. Sewall, "Dynamic Crashing Keeps Projects Moving," Computer Decisions, February 1972.

12. Berge, Claude and A. Ghouila-Houra, Programming, Games and Transportation Networks, John Wiley and Sons, Inc., 1965.

13. Berge, Claude, The Theory of Graphs and Its Applications, Translated by Alison Doig, John Wiley and Sons, Inc., 1964.

14. Berman, E. B., "Resource Allocation in a PERT Network Under Continuous Time Cost Functions," Management Science, 10, No. 4, 1964, 734-744.

15. Brand, J. D., W. L. Meyer and L. R. Shaffer, "The Resource Scheduling Problem in Construction," Construction Research Series No. 5, Department of Civil Engineering, University of Illinois, June 1964.

16. Burgess, A. R. and J. B. Killebrew, "Variations in Activity Level on a Cyclic Arrow Diagram," Journal of Industrial Engineering, Vol. 13, No. 2, Mar-Apr. 1962, 76-83.

17. Burton, Richard May, "Some Mathematical Models for the Allocation of Limited Resources to Critical Path Type Scheduling Problems," Ph.D. Thesis, Department of Business Administration, University of Illinois, 1967.

18. Busacker, R. G. and T. L. Saaty, Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill Book Co., 1965.

19. Butcher, William S., "Dynamic Programming for Project Cost-Time Curves," Journal of the Construction Division, ASCE Proceedings, Vol. 93, No. C01 (March, 1967), pp. 59-73.

20. Caterpillar Performance Handbook, Caterpillar Tractor Company, Peoria, Ill., Jan. 1976.

21. CEIR, A System Technique for Resource Allocation and Multi-Project Scheduling (RAMPS) - User Guide, Control Data Corporation, undated.

22. CEIR, A System Technique for Resource Allocation and Multi-Project Scheduling (RAMPS) - An Introduction, Control Data Corporation, undated.

23. Chapman, C. B., "The Optimal Allocation of Resources to a Variable Timetable," Operations Research Quarterly, Vol. 21, No. 1 (March, 1970), pp. 81-90.

24. Clarke, R. W., "Activity Costing - Key to Progress in Critical Path Analysis," IRE Transactions in Engineering Management, EM-9, No. 3, 1962, 132-135.

25. Cortes-Rivera, G., "Flood Control Project Planning by Mathematical Planning: A Project Expansion Approach," Ph.D. Thesis, University of Illinois, Urbana, Illinois, 1973.

26. Davis, E. D., "Networks: resource allocation," Journal of Industrial Engineering, pp. 22-32, April 1974.

27. Davis, Edward W., "Resource Allocation in Project Network Models--A Survey," Journal of Industrial Engineering, April 1966.

28. Davis, E. W. and G. E. Heidorn, "An Algorithm for Optimal Project Scheduling Under Multiple Resource Constraints," Management Science, 17, No. 12, August 1971, B803-B816.

29. Davis, E. W., An Exact Algorithm for the Multiple Constrained--Resource Project Scheduling Problem," Ph.D. Thesis, Yale University, 1968.

30. Dessouky, M. I. and E. J. Dunne, "Cost Duration Analysis with the Cut Network," AIIE Transactions, 3, No. 2, June 1971, 123-132.

31. Dessouky, M. I. and E. J. Dunne, "Proper Oriented Cut-Sets and Their Properties," Submitted to Operations Research, 1970.

32. DeWitte, L., "Manpower Leveling of PERT Networks," Data Processing for Science/Engineering, 2, No. 2, 1964, 29.

33. Dike, Sheldon H., "Project Scheduling with Resource Constraints," IEEE Transactions on Engineering Management, Vol. EM-11, Dec. 1964, pp. 155-157.

34. Dodge Construction Pricing and Scheduling Manual, McGraw-Hill, New York, N.Y.

35. Dunne, E. J., Network Theory and Project Resource Management, Ph.D. Thesis, University of Illinois, 1970.

36. Ellingsen, C. E. and P. Klovstad, "Project Planning and Control in Norwegian Shipbuilding Using Activity Oriented Network Programs," Applications of Critical Path Techniques, Edited by J. Brennan, American Elsevier Publishing Co., Inc., 1968.

37. Ellwein, L. B., "A Flexible Enumeration Scheme for Zero-One Programming," Operations Research, Vol. 22, No. 1, and Lab 1974, pp. 144-150.

38. Elmaghraby, S. E., "The One Machine Sequencing Problem with Delay Costs," Journal of Industrial Engineering, 19, No. 2, February 1968, 105-110.

39. Elmaghraby, S. E., "The Determination of Optimal Activity Question in Project Scheduling," Journal of Industrial Engineering, Vol. 19, No. 1, January 1968, pp. 48.

40. Elmaghraby, S. E., "The Determination of Optimal Activity Duration in Project Scheduling," Journal of Industrial Engineering, 19, No. 1, January 1968, 48-51.

41. Elmaghraby, S. E., "The Machine Sequencing Problem - Review and Extensions," Naval Research Logistics Quarterly, 15, No. 2, June 1968, 205-215.

42. Elmaghraby, S. E., "The Sequencing of Related Jobs," Naval Research Logistics Quarterly, 15, No. 1, March 1968, 23-32.

43. Fendley, L. G.,"Toward the Development of a Complete Multiproject Scheduling System," Journal of Industrial Engineering, Vol. 19, No. 10, (October 1968), pp. 505-515.

44. Fondahl, John W., "A Noncomputer Approach to the Critical Path Method for the Construction Industry," Department of Civil Engineering, Stanford University, Stanford, Calif., 1962.

45. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.

46. Fulkerson, D. R., "A Network Flow Computation for Project Cost Curves," Management Science, 7, No. 2, 1961, 167-178.

47. Galbreath, Robert V., "Computer Program for Leveling Resource Usage," Journal of the Construction Division, ASCE Proceedings, Vol. 91, No. C014319, (1956), pp. 107-124.

48. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17, 1969, pp. 437-454.

49. Glover, Fred and Darwin Klingman, "Real World Applications of Network Related Problems and Breakthroughs in Solving Them Efficiently," ACM Transactions on Mathematical Software, Vol. 1, No. 1, March 1975, pp. 47-55.

50. Gordon, G., System Simulation, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1969.

51. Gorenstein, Samuel, "An Algorithm for Project (Job) Sequencing with Resource Constraints," Operations Research, Vol. 20, No. 4 (July-August, 1972), pp. 835-850.

52. Halpin, D. W., "An Investigation of the Use of Simulation Networks for Modeling Construction Operations," Ph.D. Thesis, University of Illinois, Urbana, Illinois, 1973.

53. Heidari, M., V. T. Chow, and D. D. Meredith, Water Resources "Systems Analysis by Discrete Differential Dynamic Programming," Hydraulic Engineering Series No. 24, University of Illinois, Urbana, Illinois, 1971.

54. Hollander, G., "Integrated Project Control," Project Management Quarterly, April 1973.

55. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley Publishing Co., 1969.

56. Ibaraki, T., T. K. Liu, C. R. Baugh and S. Muroga, "An Implicit Enumeration Program for Zero-One Integer Programming," International Journal of Computer and Information Sciences, Vol. 1, No. 1, (March, 1972), pp. 75-92.

57. Ignall, Edward and Linus Schrage, "Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems," Operations Research, Vol. 13, No. 3, (May-June, 1965), pp. 400-412.

58. Jewell, W. S., "Divisible and Moveable Activities in Critical Path Analysis," Operations Research, Vol. 14, No. 2 (March-April, 1971), pp. 323-346.

59. Jewell, W. S., "Divisible Activities in Critical Path Analysis," Operations Research, Vol. 13, No. 5 (September-October, 1965), pp. 747-760.

60. Kahn, H., "Applications of Monte Carlo," Memorandum AECU-2359, The RAND Corporation, April, 1954.

61. Kaufmann, A., Graphs, Dynamic Programming and Finite Games, Academic Press, 1967.

62. Kelley, J. E., Jr., "Parametric Programming and the Primal-Dual Algorithm," Operations Research, 7, No. 3, 1959, 327-334.

63. Kelley, J. E., Jr., "The Critical Path Method: Resources Planning and Scheduling," Chapter 21, Industrial Scheduling, Edited by J. F. Muth and G. L. Thompson, Prentice-Hall, Inc., 1963.

64. Kelley, James E., Jr., and Morgan R. Walker, "Critical-Path Planning and Scheduling, Proceedings. Eastern Joint Computer Conference, 1959, pp. 160-173.

65. Kelley, J. F., Jr., "Critical Path Planning and Scheduling Mathematical Basis," Operations Research, Vol. 9, May-June, 1961, pp. 296-320.

66. Lambourn, S., "Resource Allocation and Multi-Project Scheduling (RAMPS) - A New Tool in Planning and Control," The Computer Journal, 5, No. 4, January 1963, 300-304.

67. Lawler, E. L. and D. E. Wood, "Branch and Bound Methods: A Survey," Operations Research, Vol. 14, No. 4, 1966, pp. 699-767.

68. Levy, F. K., G. S. Thompson and J. D. Wiest, "Multi-Ship, Multi-Shop Workload Smoothing Program," Naval Research Logistics Quarterly, 9, No. 1, 1963, 37-44.

69. Litsios, Socrates, "A Resource Allocation Problem," Operations Research, Vol. 13, No. 6 (November-December, 1965), pp. 960-988.

70. Liu, Tso-Kai, "A Code for Zero-One Integer Linear Programming by Implicit Enumeration (A Programming Manual for ILLIP)," Report No. 302, Department of Computer Science, University of Illinois, December, 1968.

71. Marganthaler, C. R., "Allocation of Constrained-Resources to Project Activities," Ph.D. Thesis, University of Illinois, 1972.

72. Martino, R. L., Critical Path Networks, MDI Publications, 1967.

73. Martino, R. L., Resources Management, MDI Publications, 1968.

74. Martino, R. L., Project Management and Control, Vol. II - Applied Operational Planning, MDI Publications, 1964.

75. Martino, R. L., Project Management and Control, Vol. III - Allocating and Scheduling Resources, MDI Publications, 1965.

76. McGee, A. A. and M. D. Markarian, "Optimum Allocation of Research/ Engineering Manpower within a Multi-Project Organizational Structure," IRE Transactions on Engineering Management, EM-9, No. 3, September 1962, 104-108.

77. Meyer, Walter L. and Louis R. Shaffer, "Extending CPM for Multiform Project Time-Cost Curves," ASCE Cons Division, Vol. 91, May 1965, pp. 45-67.

78. Meyer, W. L. and L. R. Shaffer, "Extensions of the Critical Path Method Through the Application of Integer Programming," Construction Research Series No. 2, Department of Civil Engineering, University of Illinois, July 1963.

79. Mize, Joe H., A Heuristic Scheduling Model for Multi-Project Organizations," Ph.D. Thesis, Purdue University, 1964.

80. Moder, Joseph J., and Cecil R. Phillips, Project Management with CPM and PERT, Reinhold Publishing Corporation, New York, N.Y., 1964.

81. Moshman, J., J. Johnson and M. Larson, "RAMPS - Technique for Resource Allocation and Multi-Project Scheduling," Proceedings Spring Joint Computer Conference, 1963, 23, 1963, 17-25.

82. O'Brien, James J., CPM in Construction Management, McGraw-Hill Book Co., Inc., New York, N.Y., 1965, pp. 193-198.

83. Patton, G. T., "Optimal Schedules for Resource-Constrained Projects," 36th National Meeting of ORSA, Miami Beach, Florida, August, 1968.

84. Patton, G. T., Optimal Scheduling of Resource Constrained Projects, Ph.D. Thesis, Stanford University, 1968.

85. Paulson, Boyd C., Jr., "Project Planning and Scheduling: Unified Approach," Journal of the Construction Division, ASCE Proceedings, Vol. 99 (No. CO1, July, 1973), pp. 45-49.

86. Paulson, Boyd C., Jr., "Man-Computer Concepts for Project Management," Ph.D. Thesis, Stanford University, Stanford, Ca., 1971.

87. Petrovic, R., "Optimization of Resource Allocation in Project Planning," Operations Research, 16, No. 3, 1968, 559-569.

88. Pritsker, A. A. B. and P. J. Kiviat, Simulation with GASP II, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1969.

89. Pritsker, A. A. B., L. W. Miller, R. J. Finkl, "Sequencing n Products Involving m independent Jobs on m Machines," American Institute of Industrial Engineers Transactions, Volume III, No. 1 (March 1970), pp. 49-60.

90. Pritsker, A. A. B. and W. W. Happ, "Graphical Evaluation and Review Technique," Journal of Industrial Engineering, Vol. 17, No. 5, May 1966.

91. Pritsker, A. A. B., L. J. Watters and P. M. Wolfe, "Multi-Project Scheduling with Limited Resources: A Zero-One Programming Approach," Management Science, 16, No. 1, September 1969, 237-240.

92. Richards, J. L., "An Integrated Approach to Construction Management," Ph.D. Thesis, University of Illinois, 1973.

93. Roimond, J. F., "Minimaximal Paths in Disjunctive Graphs by Direct Search," IBM Journal of Research and Development 13, 391-399 (1969).

94. Rosenbloom, R. S., "Notes on the Development of Network Models for Resource Allocation in Research and Development Models," IRE Transactions on Engineering Management, EM-11, No. 2, June 1964, 58-63.

95. Roy, B., and M. Dibon, Revue Automatisme, Feb., 1966.

96. Roy, B. and B. Sussman, "Les Problimes d'ordonnancement auec c onstraintes disjonctives," SEMA, Note D.S. No. 9 bis., décembre 1964.

97. Rubinovitch, Michael, "A Probabalistic Model for Optimal Project Scheduling," Operations Research, Vol. 20, No. 2, March-April 1972, pp. 309-317.

98. Schrage, Linus, "Solving Resource-Constrained Network Problems by Implicit Enumeration-Preemptive Case," Operations Research, Vol. 20 No. 3, May-June, 1972, pp. 668-677.

99. Shaufelberger, J. E., "A Systems Approach to the Operation of Flood Control Reservoirs," Ph.D. Thesis, University of Illinois, Urbana, Ill., 1971.

100. Shaffer, L. R., J. B. Ritter and W. L. Meyer, The Critical Path Method, McGraw-Hill Book Co., 1965.

101. Siemens, N., "A Simple CPM Time-Cost Tradeoff Algorithm," Management Science, February 1971.

102. Spivak, John, "Resource Scheduling," IBM Corporation, San Jose, Calif., June 17, 1964.

103. Sunago, Teruo, "A Method of the Optimal Scheduling for a Project with Resource Restrictions," Journal of the Operations Research Society of Japan, Vol. 12, No. 2 (March, 1970), pp. 52-64.

104. Tate, A. E., "Introduction to the Resource Allocation Problem," Paper to 2nd Critical Path Analysis Symposium, 11th June, Operational Research Society, London, 1964.

105. Tauxe, G. W., W. A. Hall, and W. W-G. Yeh, "Joint Operation of a Linked Reservoir System with Multi-state Incremental Dynamic Programming," Transactions, American Geophysical Union, Vol. 54, No. 4, p. 266, April, 1973.

106. Taylor, J. E. and J. E. Walsh, "Planning by Resource Allocation Methods," Operations Research, Vol. 12, No. 5, Sept.-Oct. 1964, pp. 693-706.

107. Thangauelu, S. R. and C. M. Shetty, "Assembly Lines Balancing by Zero-One Integer Programming," American Institute of Industrial Engineers Transactions, Vol. III, No. 1, March 1971, pp. 61-68.

108. "Uniform System for Building Construction Specifications, Data Filing and Cost Accounting," Cost Control and CPM in Construction, Associated General Contractors of America, Washington, D. C., 1968.

109. Wagner, H. W., R. J. Giglion, and R. G. Glaser, Management Science, Vol. 10, 342, 1965.

110. Waldron, James A., Fundamentals of Project Planning and Control, 371 Kings Highway West, Haddenfield, N. J., August 1963.

111. Weist, J. D., "Some Properties of Schedules for Large Projects with Limited Resources," Operations Research, Vol. 12, No. 3 (1964), pp. 395-418.

112. Weist, J. D., "A Heuristic Model for Scheduling Large Projects with Limited Resources," Management Science, Vol. 13, No. 6, (February, 1967), pp. B359-B377.

113. Zaloom, V., "On the Resource Constrained Project Scheduling Problem," AIIE Transactions, Vol. III, No. 4, Dec. 71, pp. 302-305.

114. Zangwill, W. I., Minimum Concave Cost Flows in Certain Networks, Center for Research in Mgt. Sco., U. of Cal., July 1967.

115. Zimmerman, L. S. and L. R. Shaffer, "A Network Approach to Resource Scheduling," Construction Research Series No. 10, Department of Civil Engineering, University of Illinois, 1967.

VITA

Robert Lowell Preston was born January 17, 1937 at Madisonville, Kentucky. He attended Madisonville High School until entering the United States Naval Academy at Annapolis, Maryland. After graduating from Annapolis in June of 1959, he entered the U. S. Navy Civil Engineer Corps as an Ensign and has subsequently received the Bachelor of Civil Engineering from Rensselaer Polytechnic Institute and the Master of Science in Civil Engineering from Georgia Institute of Technology. Since entering the Navy Civil Engineer Corps, Commander Preston has been continuously assigned construction management responsibilities in Japan, Southeast Asia, Europe and the United States. Assignments have included responsibility for construction of such facilities as aircraft hangers, permanent runways for jet aircraft, dry dock facilities, roads, highways, barracks, waterfront structures, multi-story concrete structures, chapels, theaters, clubs, bridges, petroleum farms, power generating stations, cold storage facilities and quarry development. Most notable assignments have been the construction of 582 units of single and multi-family housing, total responsibility for maintenance and new construction of the multi-command military installation at Lakehurst, N. J. and direct responsibility for all construction accomplished by Naval Construction Forces in the Carribean, European and Mediterranean geographical areas. Commander Preston has received the Bronze Star for exceptional management accomplishments and the Naval

Facilities Engineering Command "Model Public Works" award for achievement in Civil Engineering management. He is a member of the American Society of Civil Engineers, Chi Epsilon honor society and a registered professional engineer in New Jersey and Massachusetts. In 1973 Commander Preston was selected for additional graduate study and was assigned to the Georgia Institute of Technology to pursue a Ph.D. in Civil Engineering.